RasPi

DESIGN
BUILD
CODE

20

3D PRINT A CASE

HACK MINECRAFT

Plus

Monitor your network with Nagios

Welcome

Grab your digital pickaxe because we have a lot of material to get through this month! Our big Minecraft feature will show you how to turn your Raspberry Pi into a dedicated console for the game, and we've also made a cool case for it that you can 3D-print. Once you're set up, get ready to hack – starting with a simple hide-and-seek game, we'll teach you how to write Python scripts that can extend the possibilities of your Minecraft world. Then, we'll make a few more mods and assign them to physical buttons that you can stitch into the fingers of a spare glove, so you can deploy power moves in an exciting chase game – TNT walls and lava pits will be right at your fingertips. Enjoy yourself!

Gavin Thomas

Editor

Get inspired

Discover the RasPi community's best projects

Expert advice

Got a question? Get in touch and we'll give you a hand

Easy-to-follow guides

Learn to make and code gadgets with Raspberry Pi

From the makers of

LinuxUser & Developer

Join the conversation at...

@linuxusermag | Linux User & Developer | RasPi@imagine-publishing.co.uk

Contents

Build a Raspberry Pi Minecraft console

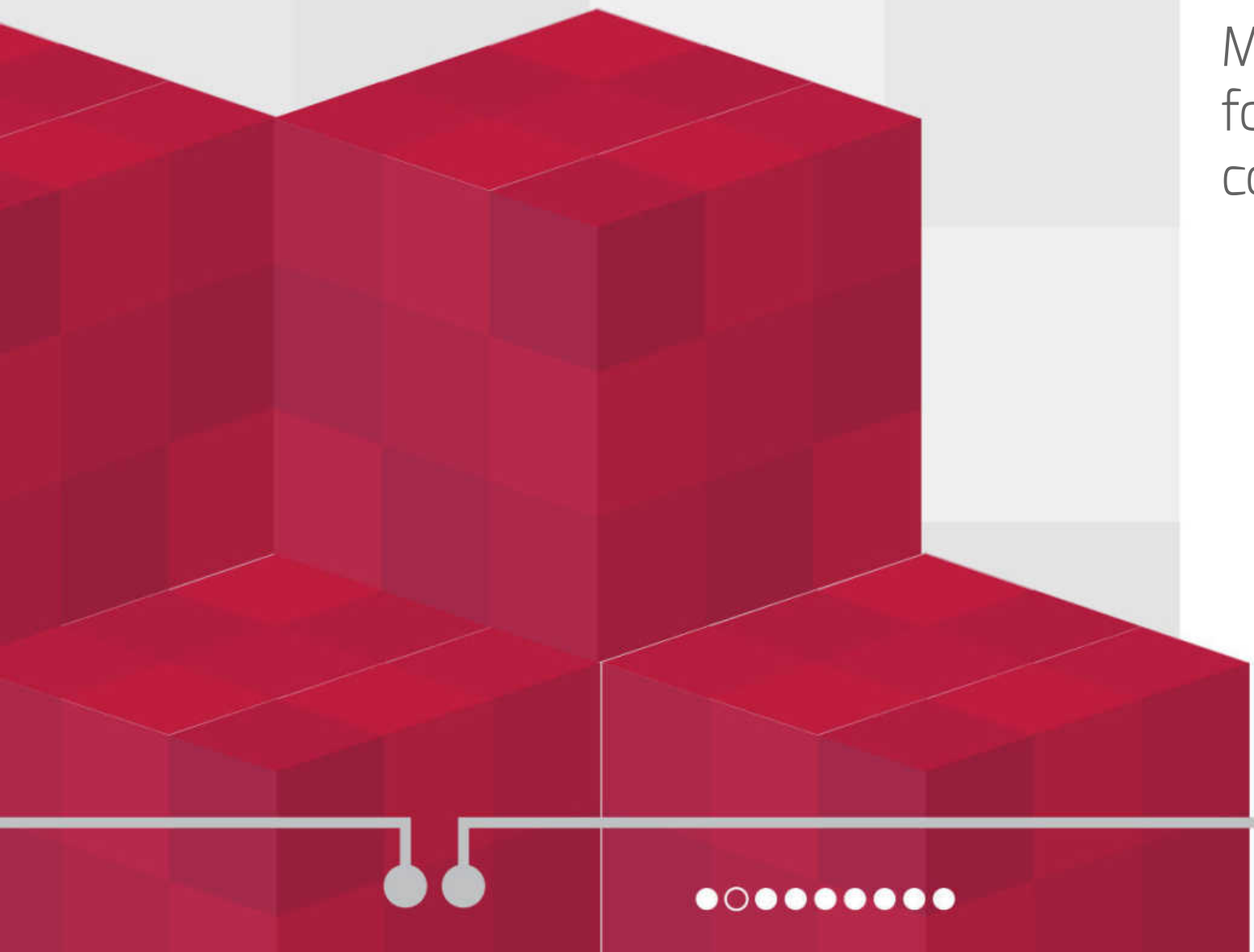
Create a dedicated, Pi-powered games console for playing Minecraft and learning how to program, too

"Using a Raspberry Pi is also the cheapest way to get a fully-functional version of Minecraft up onto your TV"

Minecraft means many things to many people, and to Raspberry Pi users it's supposed to mean education. Not everyone knows, though, that you can still have fun and play Minecraft as you normally would.

Using a Raspberry Pi is also the cheapest way to get a fully-functional version of Minecraft up onto your TV. However, in its normal state, just being on a TV screen isn't the end of it. Using all the features and functions of the Pi, we can take it to a state more fitting of a TV by making it into a hackable, moddable Minecraft console.

Over the next few pages, we will show you how to set it up in terms of both software and hardware, how to add and configure a game controller to make it a bit better for TV use, and we'll even give you an example script showing how to mod it, so you can learn how to completely transform your game with just a few lines of code. We've also got a CAD model for you to 3D-print – a custom Minecraft case for your new console! Now, it's time to get building, so swipe forward to start.



THE PROJECT ESSENTIALS

Raspberry Pi 2

Latest Raspbian image
raspberrypi.org/downloads

Minecraft Pi Edition
http://pi.minecraft.net

Raspberry Pi case

USB game controller
(PS3 preferable)

"We've also got a CAD model for you to 3D-print – a custom Minecraft case for your new console!"

01 Choose your Raspberry Pi

Before we start anything, everything we plan to do in this tutorial will work on all Raspberry Pi Model Bs with at least 512 MB of RAM. However, Minecraft: Pi Edition can chug a little on the original Model Bs, so we suggest getting a Raspberry Pi 2 to get the most out of this tutorial.

02 Prepare your Raspberry Pi

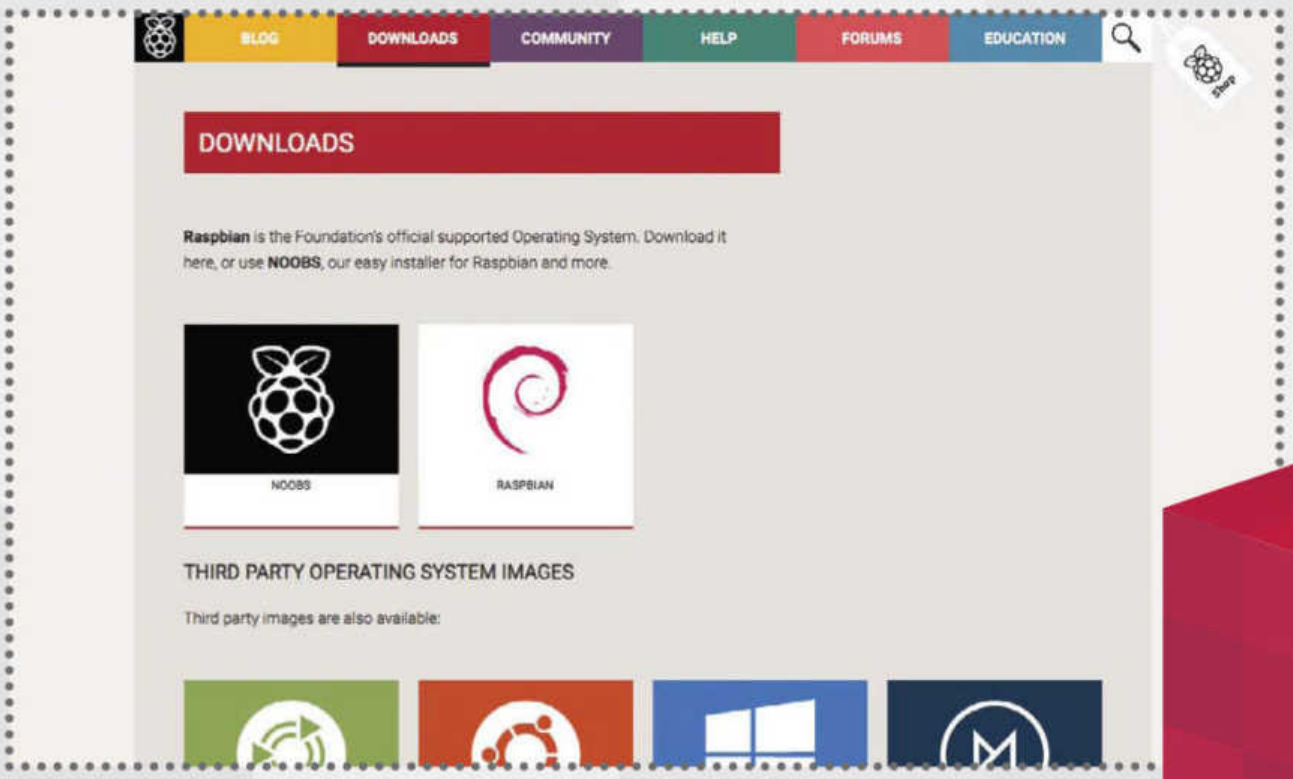
Minecraft: Pi Edition currently works on Raspbian. We recommend you install a fresh version of Raspbian, but if you already have an SD card with it on, the very least you should do is:

```
sudo apt-get update && sudo apt-get upgrade
```

03 Prepare Minecraft

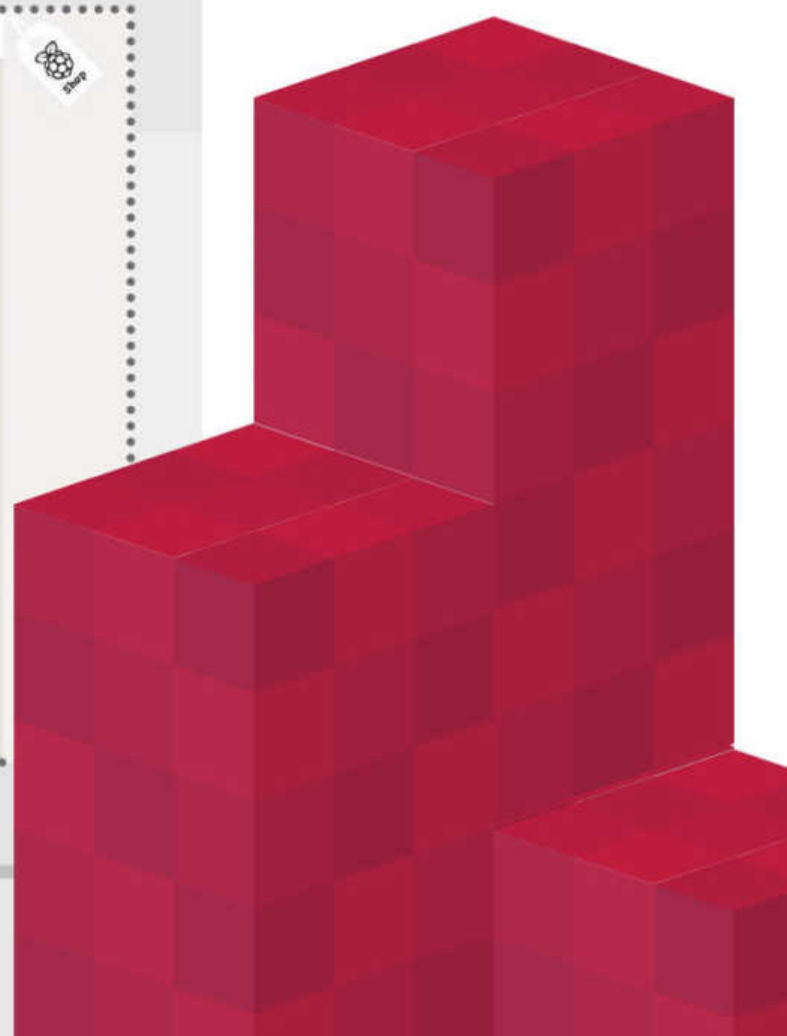
If you've installed Raspbian from scratch, Minecraft is actually already installed – go to the Menu and look under Games to find it there ready. If you've just updated your version of Raspbian, you can install it from the repos with:

```
$ sudo apt-get install minecraft-pi
```



Updates to Pi Edition?

Minecraft: Pi Edition hasn't received an update for a long time, and unfortunately there is no indication that it will see one any time soon. The original developers have since moved on to other projects, and although Microsoft now owns Minecraft, it has recently focused on the launch of Minecraft: Education Edition instead (**http://education.minecraft.net**).



04 Test it out

If you've had to install Minecraft, it's best just to check that it works first. Launch the desktop, if you're not already in it, with **startx** and start Minecraft from the Menu. Minecraft: Pi Edition is quite limited in what it lets you do, but it does make room for modding.

05 X setup

If you have a fresh Raspbian install and/or you have your install launch into the command line, you need to set it to load into the desktop. If you're still in the desktop, open up the terminal and type **raspi-config**. Go to Enable Boot to Desktop and choose Desktop.

06 Set up Python

While we're doing set up bits, we might as well modify Mincecraft using Python for a later part of the tutorial. Open up the terminal and use:

```
$ cp /opt/minecraft-pi/api/python/mcpi ~/
minecraft/
```

07 Minecraft at startup

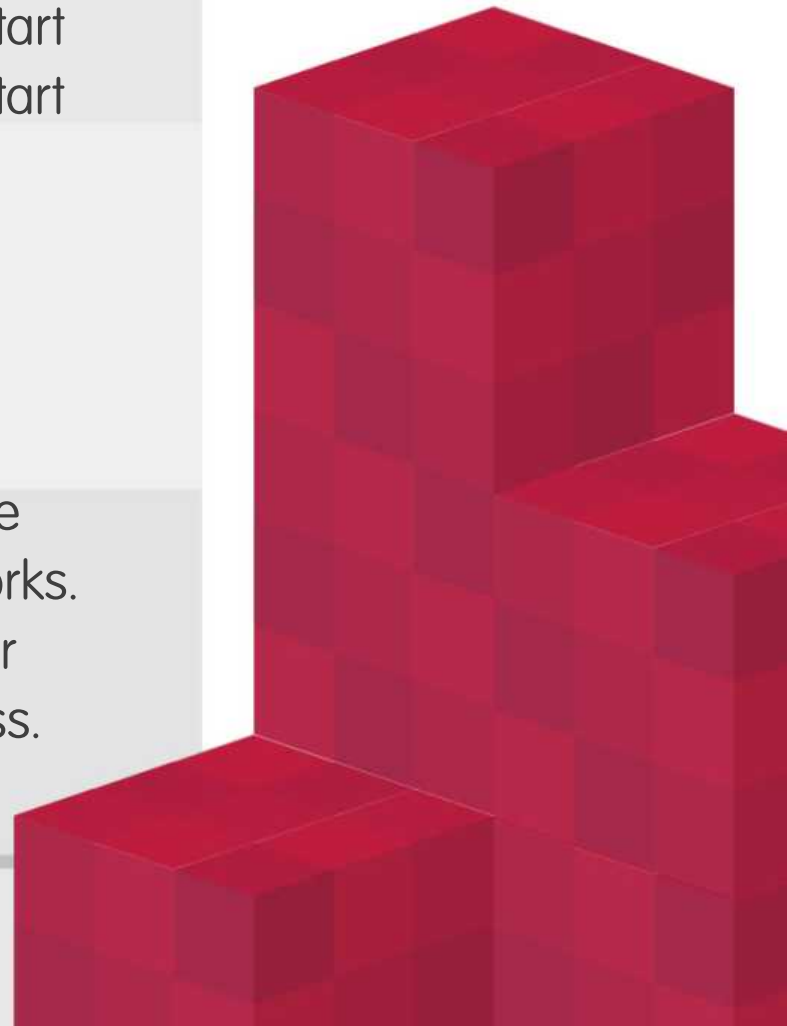
For this to work as a console, we'll need it to launch into Minecraft when it turns on. We can make it autostart by going into the terminal and opening the autostart options by typing:

```
$ sudo nano /etc/xdg/lxsession/LXDE-pi/
autostart
```

08 Autostart language

In here, you just need to add @minecraft-pi on the bottom line, save it and reboot to make sure it works. This is a good thing to know if you also want other programs to launch as part of the boot-up process.

"If you've installed Raspbian from scratch, Minecraft is already installed – go to the Menu and look under Games"



09 Turn off

For now, we can use the mouse and keyboard to shut down the Pi in the normal way, but in the future you'll have to start turning it off by physically removing power. As long as you've exited the Minecraft world and saved, that should be fine.

10 The correct case

In this scenario, we're hooking this Raspberry Pi up to a TV, which means it needs a case so that there's less chance of damage to the components from dust or static. There are many good cases you can get – we quite like using the Pimoroni Pibow because you can mount it to the back of the TV, although the official case is also great. Alternatively, you could get really creative and 3D-print your own case, as you can see below. Check out the boxout just to the right.

Left Our Minecraft case has a sliding panel in the base to open it up

3D-print a case

Aaron Hicks at Solid Technologies designed this Minecraft case for the Raspberry Pi and shared it on GrabCAD. We've uploaded a slightly modified version of the file for you (with better fitting) over at **http://bit.ly/1opjZck**. All you need to do is send the STL file to a 3D printing service – many high street printing shops have at least a MakerBot these days – and it should only cost around £15.

11 Find the right power supply

Getting power to the Raspberry Pi 2 so that it runs properly can be tricky if you're using a spare USB port or a mobile phone charger – the former will be underpowered and the latter is not always powerful enough. Make sure you get a 2A supply, like the official Raspberry Pi one.

12 Go wireless

We understand that not everyone has an Ethernet cable near their TV, so it may be a good idea to invest in a Wi-Fi adapter instead. We recommend the official module, and there is a great list of compatible Wi-Fi adapters at: **http://elinux.org/RPi_VerifiedPeripherals**.

13 Mouse and keyboard

Now that we have the Raspberry Pi ready to be hooked up, you should look at your controller situation – do you want to be limited by the wires or should you get a wireless solution instead? We will cover controller solutions over the page, but it's worth considering now.

14 Get ready for SSH

It will be easier to create and apply scripts to Minecraft by uploading them via the network rather than doing it straight on the Pi. In the terminal, find out what the IP address is by using **ifconfig**, and then you can access the Pi in the terminal of another networked computer using the following:

```
ssh pi@[IP address]
```

15 Have a play

At this stage, what we have built is a fully-functional Minecraft console. Now, at this point you could start

"Getting power to the Raspberry Pi 2 so that it runs properly can be tricky if you're using a spare USB port"

Left The PS3 controller is very well supported in Linux, so getting it running with Minecraft is a cinch

playing if you so wish and you don't need to add a controller. You can swipe three pages forward now if you want to begin learning how to mod your Minecraft and do a bit more with it to suit your needs. However, if you do want to add controller support then carry on and take a look at the next step.

16 Add controller support

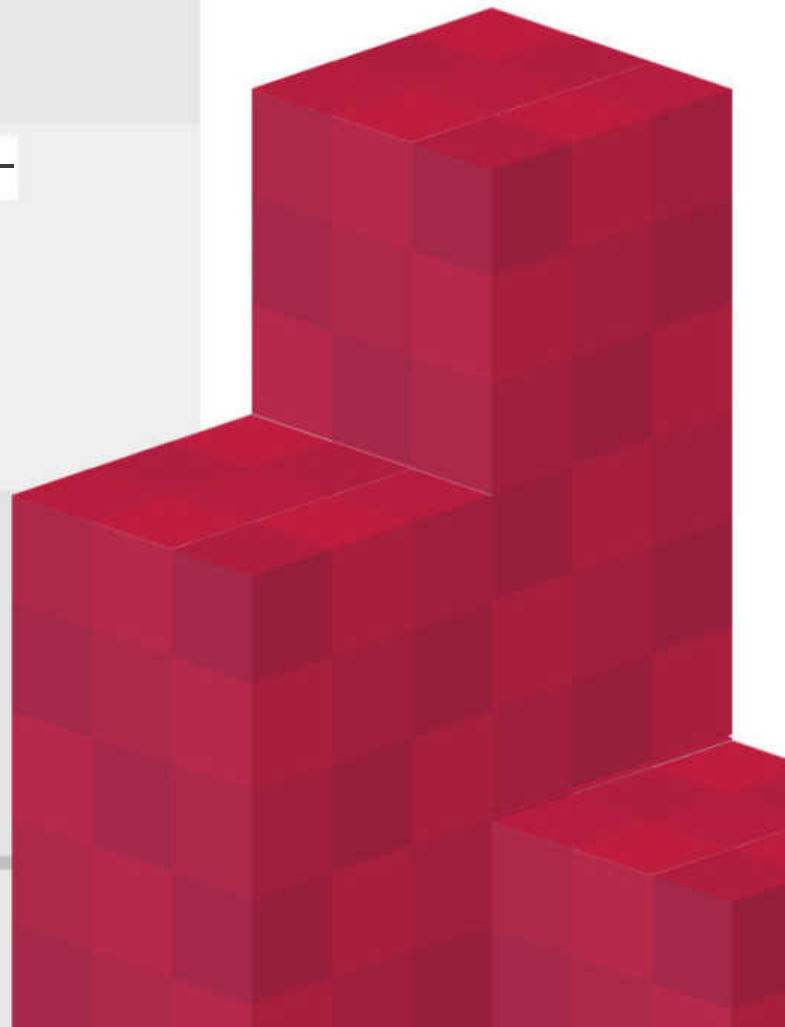
Make sure the controller input functions are installed on the Raspberry Pi. To do this, ssh into the Raspberry Pi like we did in Step 14 (where 'raspberry' is the password) and install the following package:

```
$ sudo apt-get install xserver-xorg-input-joystick
```

17 Controller mapping

We have a controller map for the PS3 controller that you can download straight to your Pi, and with a bit of tweaking can fit most USB controllers as well. Go to the controller configuration folder with:

```
$ cd /usr/share/X11/xorg.conf.d/
```



18 Replace the controller mapping

Remove the current joystick controls by using **sudo rm 50-joystick.conf** and then replace them by downloading a custom configuration using:

```
$ sudo wget https://github.com/tommybobbins/PS3-SixAxis-RaspberryPi/blob/master/50-joystick.conf
```

19 Reboot to use

After a reboot to make sure everything's working, you should be able to control the mouse input on the console. R2 and L2 are the normal mouse clicks and can be used to navigate the Minecraft menu to access the game. The full controls layout is on the next page.

20 Go full-screen

You may have noticed that Minecraft is running in a window – you can click the full-screen button to make it fill the screen, but you then heavily limit your mouse control. Thanks to the controller, you can get around that – just make sure you use the sticks for movement and the d-pad for selecting items in the inventory.

Xbox controllers

Unfortunately, Xbox 360 controllers work slightly differently with Linux. As they use their own drivers that are separate to the normal joystick drivers we used for the PS3 pad and other USB controllers, a 360 controller doesn't work as a mouse and is harder to assign specific functions to. This makes it tricky to use in a situation such as this.

Left Full-screen Minecraft Pi is possible if you use a controller with it

Controls

Here's the full layout of the buttons used by the PS3 controller by default – you can change them in the script that you download in Step 18

Mod your Minecraft

Expand the possibilities of your Minecraft world by hacking it with a few simple Python scripts

We can make Minecraft react custom Python scripts by using the API that comes with Minecraft: Pi Edition – it's what we moved to the home folder earlier on. Now's a good time to test it – we can do this remotely via SSH. Just **cd** into the Minecraft folder in the home directory we made, and use **nano test.py** to create our test file. Add the following:

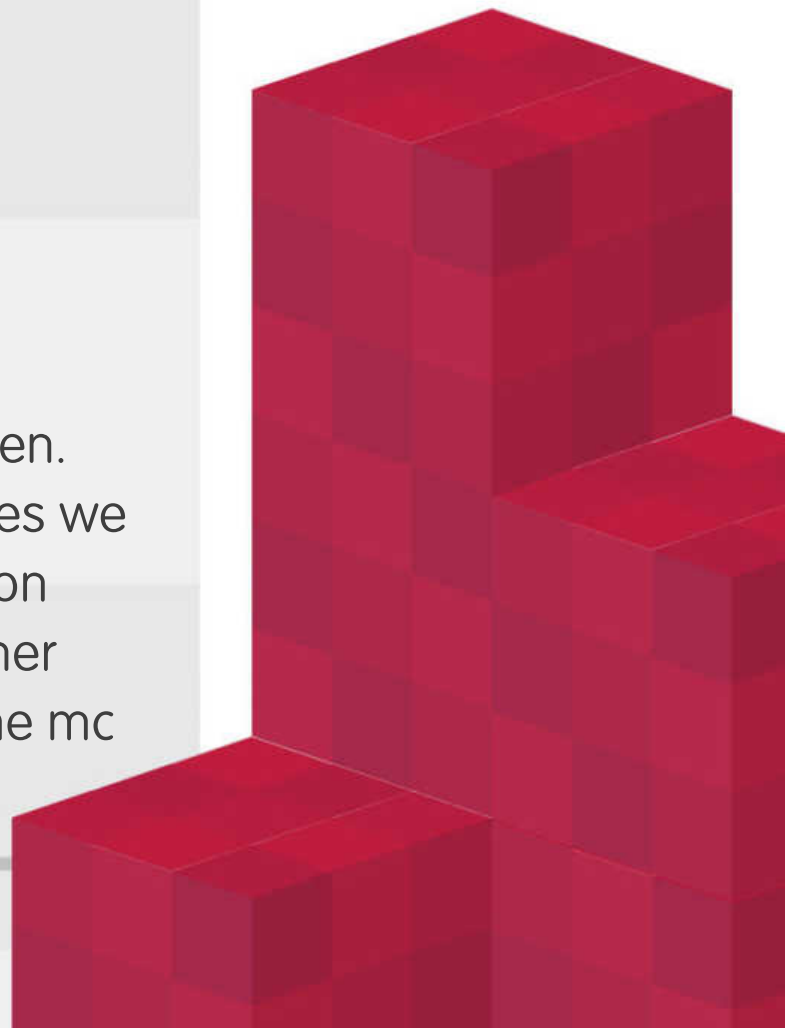
```
from mcpi.minecraft import Minecraft
from mcpi import block
from mcpi.vec3 import Vec3
mc = Minecraft.create()
mc.postToChat("Hello, Minecraft!")
```

Save it, and then run it with:

```
$ python test.py
```

In-game, "Hello, Minecraft!" should pop up on-screen. The code imports the Minecraft function from the files we moved earlier, which allows us to actually use Python to interact with Minecraft, along with the various other functions and modules imported. We then create the mc

> "We program Minecraft to react in Python using the API that comes with Minecraft Pi – it's what we moved to the home folder earlier"



Left Here's a screenshot of our hidden diamond and the postToChat function in action

instance that will allow us to actually post to Minecraft using the postToChat function. There are many ways you can interact with Minecraft in this way – placing blocks that follow the player, creating entire structures and giving them random properties as they're spawned as well. There are very few limits to what you can do with the Python code, and you can check out more projects here: **https://mcpipy.wordpress.com.**

Starting on the next page, we have a full listing for a hide and seek game that expands on the kind of code we're using here, where the player must find a diamond hidden in the level, with the game telling you whether you're hotter or colder. You can write it out from scratch or download it to your Pi using the following commands:

```
$ wget http://www.linuxuser.co.uk/tutorialfiles/
   Issue134/ProgramMinecraftPi.zip
$ unzip ProgramMinecraftPi.zip
$ cp Program\ MinecraftPi/hide_and_Seek.
   py ~/minecraft
```

To do it from scratch, just type up these sections in order:

Import Here we're importing the necessary modules and APIs to program Minecraft. Most importantly are the files in the mcpi folder that we copied earlier:

```
from mcpi.minecraft import Minecraft
from mcpi import block
from mcpi.vec3 import Vec3
from time import sleep, time
import random, math
```

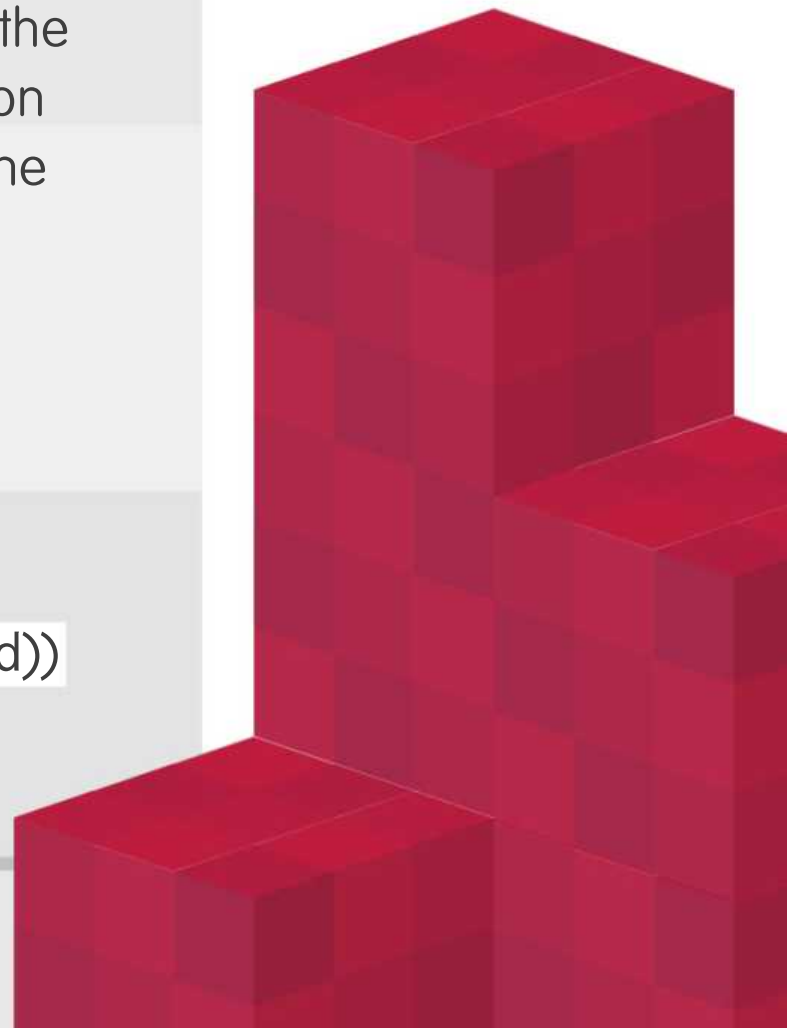
Locate We connect to Minecraft with the first line, and then we find the player's position and round it up to an integer, ensuring we have x, y and z coordinates:

```
mc = Minecraft.create()
playerPos = mc.player.getPos()

def roundVec3(vec3):
  return Vec3(int(vec3.x), int(vec3.y), int(vec3.z))
```

Range finding Calculate the distance between the player and diamond. This is done in intervals later on in the code, and just compares the coordinates of the positions together:

```
def distanceBetweenPoints(point1, point2):
  xd = point2.x - point1.x
  yd = point2.y - point1.y
  zd = point2.z - point1.z
  return math.sqrt((xd*xd) + (yd*yd) + (zd*zd))
```

"We connect to Minecraft with the first line, and then we find the player's position and round it up to an integer"



Creation Next, create a random position for the diamond within 50 blocks of the player position that was found earlier:

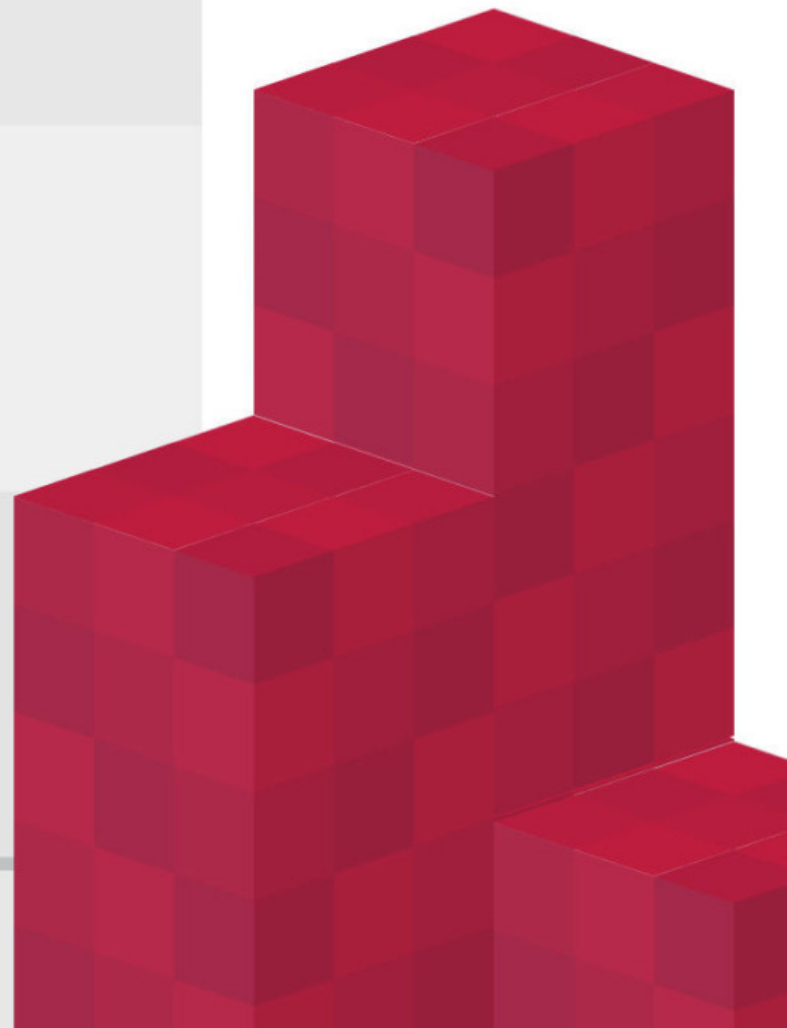
```
def random_block():
    randomBlockPos = roundVec3(playerPos)
    randomBlockPos.x = random.randrange(randomBlockPos.x - 50,
randomBlockPos.x + 50)
    randomBlockPos.y = random.randrange(randomBlockPos.y - 5,
randomBlockPos.y + 5)
    randomBlockPos.z = random.randrange(randomBlockPos.z - 50,
randomBlockPos.z + 50)
    return randomBlockPos
```

Start This is the main loop that actually starts the game. It asks to get the position of the player to start each loop.

```
def main():
    global lastPlayerPos, playerPos
    seeking = True
    lastPlayerPos = playerPos
```

Notification This part sets the block in the environment and pushes a message using postToChat to the Minecraft instance to let the player know that the mini-game has started.

```
    randomBlockPos = random_block()
    mc.setBlock(randomBlockPos, block.DIAMOND_
BLOCK)
    mc.postToChat("A diamond has been hidden
- go find!")
```



Checking We start timing the player with timeStarted, and set the last distance between the player and the block. Now we begin the massive while loop that checks the distance between the changing player position and the fixed diamond. If the player is within two blocks of the diamond, it means they have found the block and it ends the loop.

```
  lastDistanceFromBlock = distanceBetweenPoints(randomBlockPos,
lastPlayerPos)
  timeStarted = time()
  while seeking:

    playerPos = mc.player.getPos()

    if lastPlayerPos != playerPos:
      distanceFromBlock = distanceBetweenPoints(randomBlockPos,
playerPos)
      if distanceFromBlock < 2:
        seeking = False
```

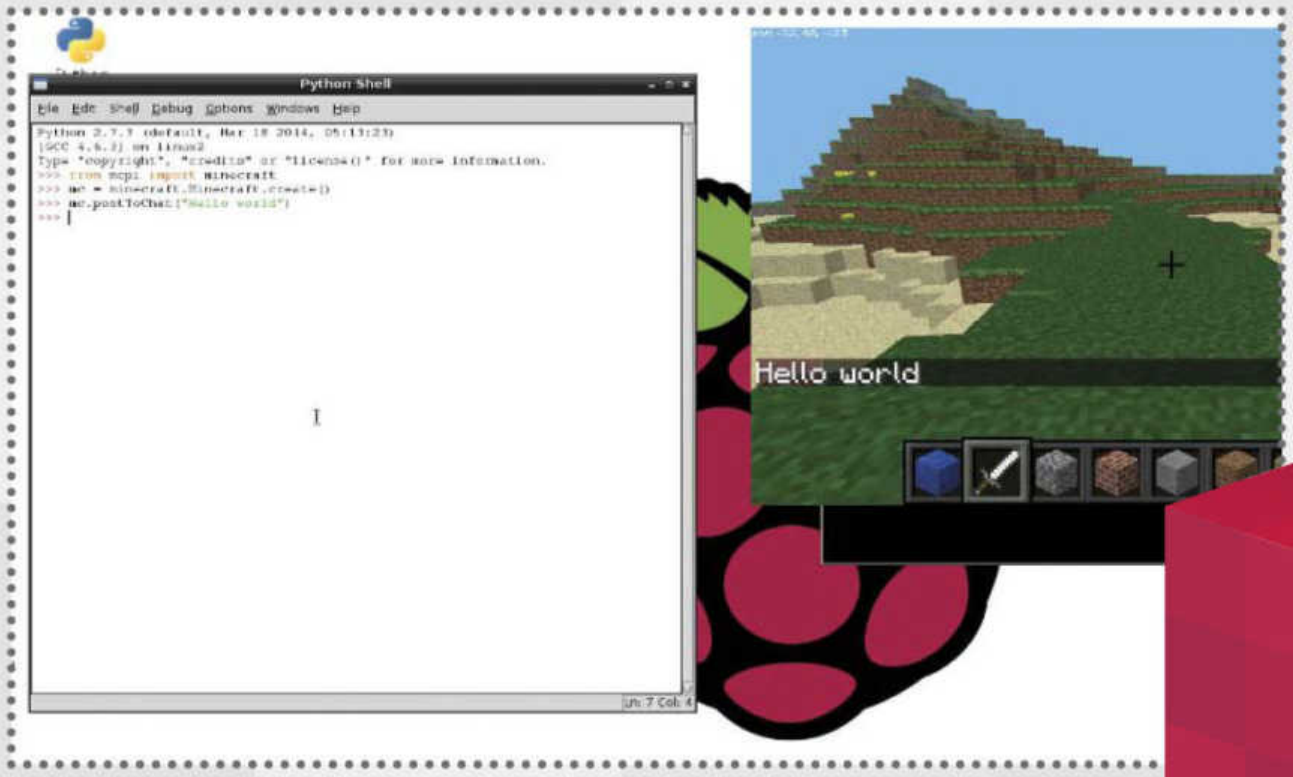
Message writing If you're two or more blocks away from the diamond, it will tell you whether you're nearer or farther away than your last position check. It does this by comparing the last and new position distance – if it's the same, a quirk in Python means it says you're colder. Once it's done this, it saves your current position as the last position.

```
      else:
        if distanceFromBlock < lastDistanceFromBlock:
          mc.postToChat("Warmer " + str(int(distanceFromBlock)) + "
blocks away")
```

```
        if distanceFromBlock > lastDistanceFromBlock:
          mc.postToChat("Colder " + str(int(distanceFromBlock)) + "
blocks away")

      lastDistanceFromBlock = distanceFromBlock

    sleep(2)
```

Success Our script takes a two-second break before updating the next position using the sleep function. If the loop has been broken, it tallies up your time and lets you know how long it was before you found the diamond. Finally, the last bit then tells Python to start the script at the main function.

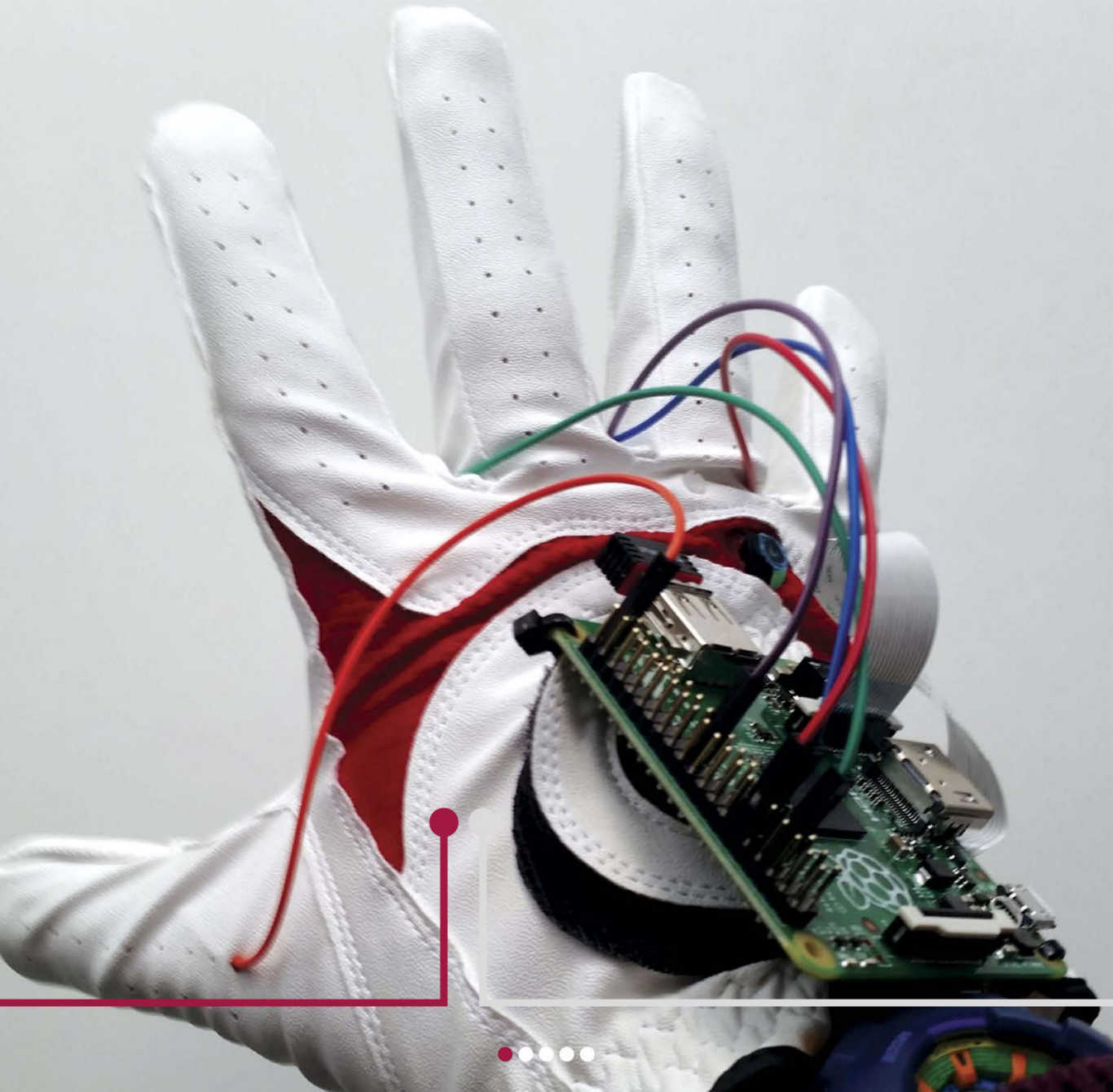
```
  timeTaken = time() - timeStarted
  mc.postToChat("Well done - " + str(int(timeTaken)) + " seconds to
find the diamond")

if __name__ == "__main__":
  main()
```



Left Want more Minecraft hacks? The Pi Foundation has you totally covered: **http://bit.ly/1TYUWIf**

Pi Glove 2

Dan Aldred's new home help module controls lights, sends texts and can read out the text on photographs



What physical modifications have you made since we last spoke?

The glove is more portable – previously, the Raspberry Pi was in the wearer's pocket and you had the long CAT5 cables attached the glove. This has all been stripped back and several steps enabled this. Firstly, was the use of fabric clothes poppers to replace the tactile switches. These are metal and when one makes contact with a ground popper on the thumb, it creates the circuit. It has meant that the same functionality is achieved with five wires as opposed to the previous ten wires. Secondly, I have moved from a B+ model to the A+ model, which has meant that the Raspberry Pi is now small enough to be mounted onto the glove itself. Now the wires only need to run from the fingertip to the wrist. The camera module is also embedded within the glove. The lens is exposed through the glove but the rest of the camera is now housed within the fabric of the glove. You have to be a little bit more careful when you take the glove off, but the overall pay-off is that the glove is now lighter and more compact. The power comes from a small USB mobile phone charger which gives about six hours running time, depending on how much you use it for.

What new functions does the rebuilt glove have?

It was always the plan to develop the Pi Glove with 'modules' for various uses, starting with home assistance. Imagine if you did or maybe do struggle with a disability and you wake up in the night – the Pi Glove now enables you to check the time, which is read out, and a light can be turned on with a simple finger movement. If required, an emergency text can be sent to a carer, family member or the provider of other medical assistance. The fourth button enables the Pi camera, which takes a picture of a sign, for

Dan Aldred is a Raspberry Pi Certified Educator and a Lead School teacher for Computing At School. He recently led a winning team of the Astro Pi secondary school contest. Check out his website over at: **http://tecoed.co.uk**.

example. OCR is then used to recognise and store the text, which is then read back to you.

I decided to add the Pi camera around the back of the hand area – this linked in well, enabling a more mobile use of the camera; it can now be positioned in a direction that the user wants, is more accessible and you could even take a selfie! The main reason for this change was to enable 'on the fly' optical character recognition. I installed a Python OCR library and, combining this with the image taken from the Pi camera, the software can identify the text within the picture. This text is then fed back to the console and easy-Speak reads out the text. I tried various file formats – JPG seemed to work well. Also, changing the picture to black and white, in order to pick up detail and differentiate between the text, had improved results. There were issues with it not identifying text in low light, and also if the text was the wrong way round or upside down. Finally, changing the saturation and increasing the sharpness produced usable results.

If you like

To learn more about the redesigned Raspberry Pi Glove and the current home help module, check out Dan's YouTube video (**bit.ly/1PFV78I**) and the project write-up (**bit.ly/19xgQyC**).

Left For the next version, Dan may add a palm button for a hierarchical menu structure to navigate the functions on each finger-button

The emergency text on the second button makes use of Twilio, the web-based communication API, which enables the user to send a pre-written message requesting assistance. This could be adopted by others, such as the police or fire brigade, for use in dangerous situations. The current time is also added to the text.

To turn the lights on I used an add-on board by Energenie. Once installed you can use it to control up to four simple Energenie radio-controlled sockets independently, using a small program. The add-on board connects directly to the GPIO, which can be controlled as either input or output lines under your software control. A Python library is also available to program the sockets. I simply mapped the 'on' state to the click of the button and it turned the light on – from your fingertips!

Are you currently developing any new Pi Glove modules for the future?

The current module I am working on is fitness-related and will let the wearer take their heart rate, change their music and upload the details of a run to their social media site. This will be on the fly with no need to stop running or whatever sporting activity you are doing. Just touch the buttons and your workout music changes, or your heart rate is read and then converted to a string and read back to you through your headphones. I find that current apps on phones and watches disrupt your workout – I don't want to have to stop running to check my pulse or change the music.

I was looking at the idea of linking the glove functionally to a smartphone but this, I feel, would be moving away from the original aim, which was to remove the cumbersomeness of the phone – having to unlock it, load

Further reading

If you're interested in setting up optical character recognition for your own sign-reading Python projects, check out Dan's guide over at TeCoEd (Teaching Computing Education): **www.tecoed.co.uk/python-ocr.html.**

an app, wait for the camera to prepare itself. The glove enables you to click and go.

What would you like to do for the third iteration of Project New York?

I was introduced to the Micro Python Pyboard (**https://micropython.org/live**), which has a range of features built-in and is smaller, making it more suitable as a wearable component. The Micro Python board is a small electronic circuit board that runs Micro Python on the bare metal, and gives you a low-level Python operating system that can be used to control all kinds of different electronic projects.

The battery size is also an area that could be reduced – I am looking into this. The smaller these components are, the more natural the glove will feel.

Below Now he's done the social media (**RasPi** #15) and the home help modules, Dan will make a start on the fitness module

Make a Minecraft power move glove

Create a piece of wearable tech with power moves assigned to each button to enhance your game

Many of you will be avid fans of the game Minecraft. In schools it is fast becoming a motivational teaching and learning tool, useful in areas such as programming, creating logic gates and setting up a network. This project is framed around creating a simple networked Minecraft game where one player chases the other and tries to hit the block they are standing on. The real hack is programming a 'power glove' that enables you to assign power moves to each finger button. These powers can then be deployed and used to slow down the other player and get yourself out of sticky situations – see the video at **http://bit.ly/1CQSmHS**! The real beauty of this hack is that you can then create and customise your own power moves. The possibilities are endless, limited only by your imagination. If you're confident with GPIO pins and setting up buttons, jump straight to Step 8.

THE PROJECT ESSENTIALS

4 x 6mm tactile buttons

Female-to-female jumper wires

Terminal blocks

Spare glove

Router

2x CAT5 cables

01 Update the Raspberry Pi

This project is designed for the Raspberry Pi 2, which requires the updated operating system, although it is compatible with the Raspberry Pi B+, too. First ensure that your software is up to date – open the LX Terminal and type:

```
sudo apt-get update
sudo apt-get dist-upgrade
sudo apt-get install raspberrypi-ui-mods
```

02 Connect a button to the Raspberry Pi

Take one of the buttons and connect a wire to each contact, then take the other two ends and connect to the Pi. You may find this easier using a female-to-female

"The real beauty of this hack is that you can then create and customise your own power moves. The possibilities are endless, limited only by your imagination"

connector. To set up a test button, use GPIO pin 17 – this is physical pin number 11 on the board. The other wire connects to a ground pin, shown by a minus sign (the pin above GPIO pin 17 is a ground pin).

03 Test that the button works

Use this test code to ensure the button is functioning correctly. Once it is, the same setup method can be used throughout this project. To ensure the buttons are responsive, use the pull-up resistor with the code GPIO. PUD_UP – this will ensure that multiple touches aren't registered on each button. Using Python 2.8, type in the code below, then save and run. If working correctly, it will return the message 'Button works'.

```
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)

GPIO.cleanup()
GPIO.setup(17, GPIO.IN, GPIO.PUD_UP)

while True:
    if GPIO.input(17) == 0:
        print "Button works"
```

04 Create a power move

Now to create your first power move. The glove and code are structured in such a way that once the basic code template is set up, you can create your own power moves and assign them to each button, keeping both your ideas and your gameplay fresh. The first power move you will program enables you to

"Once the basic code template is set up, you can create your own power moves and assign them to each button"

Above Some might consider the use of explosives in a simple game of catch to be cheating. Meh

place a wall of TNT between you and the player trying to catch you. They have a choice to go around the wall or blow it up, but it will slow them down. In a new Python window, type the code below and save the program into the home folder:

```
import time
def Firewall():
    mc.postToChat("Firewall Placed")
    TNT = 46,1
    x, y, z = mc.player.getPos()
    mc.setBlocks(x-6, y, z+2, x+6, y+10,
z+3, TNT)
    time.sleep(10)

while True:
    Firewall()
```

05 Open Minecraft

The updated version of the Raspbian OS comes with Minecraft pre-installed, it can be found under Menu>Games – so load it up. If you have used the Minecraft: Pi Edition before you will be aware that it runs better in a smaller-sized window, so don't make it full screen. You may prefer to adjust and arrange each window side-by-side in order to enable you to view both the Python code and the Minecraft game at the same time.

06 Engage the firewall

Start a new Minecraft game, then go to the Python program and press F5 to run it. You may need to press the Tab key to release the mouse from the Minecraft window. The program will place a 12 x 10 wall of TNT behind the player every ten seconds – you can blow them up but the Pi might lag!

07 Assign the power move to the button

Now you have a working button and a power move, you can combine these two together. Basically, when you press the button the power move will deploy. Once this is working you can then set up the other three buttons with the same method. Open a new Python window and type in the code below – save the file in the home folder. Start a Minecraft game as shown in Step 6 and then run the new program. Every time you press the button, you will place a TNT firewall.

```
import time
from mcpi import minecraft
mc = minecraft.Minecraft.create()
```

GPIO pins

GPIO pins are a physical interface between the Pi and the outside world. At the simplest level, you can think of them as switches that you can turn on or off (input) or that the Pi can turn on or off (output). The GPIO.BCM option means that you are referring to the pins by the "Broadcom SOC channel" number. GPIO.BOARD specifies that you are referring to the pins by the number of the pin and the plug – the numbers printed on the board.

```
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)

GPIO.setup(17, GPIO.IN, GPIO.PUD_UP)
#11 on the BOARD

def Firewall():
    mc.postToChat("Firewall Placed")
    TNT = 46,1
    x, y, z = mc.player.getPos()
    mc.setBlocks(x-6, y, z-2, x+6, y+10,
z-1, TNT)

while True:
        if GPIO.input(17) == 0:
            Firewall()
```

"If the firewall power move builds on each click of the button, the connections and buttons are working and you're ready to attach the buttons to the glove"

08 Set up further buttons

Once you have one button working the next step is to set up the other three using the same method from Step 2. Take the button and connect the two wires to either side. At this point you can add all three buttons and test them individually for connectivity. Connect each set of button wires to the GPIO 17 and run the firewall program. If the firewall power move builds on each click of the button, the connections and buttons are working and you're ready to attach the buttons to the glove.

09 Create the glove

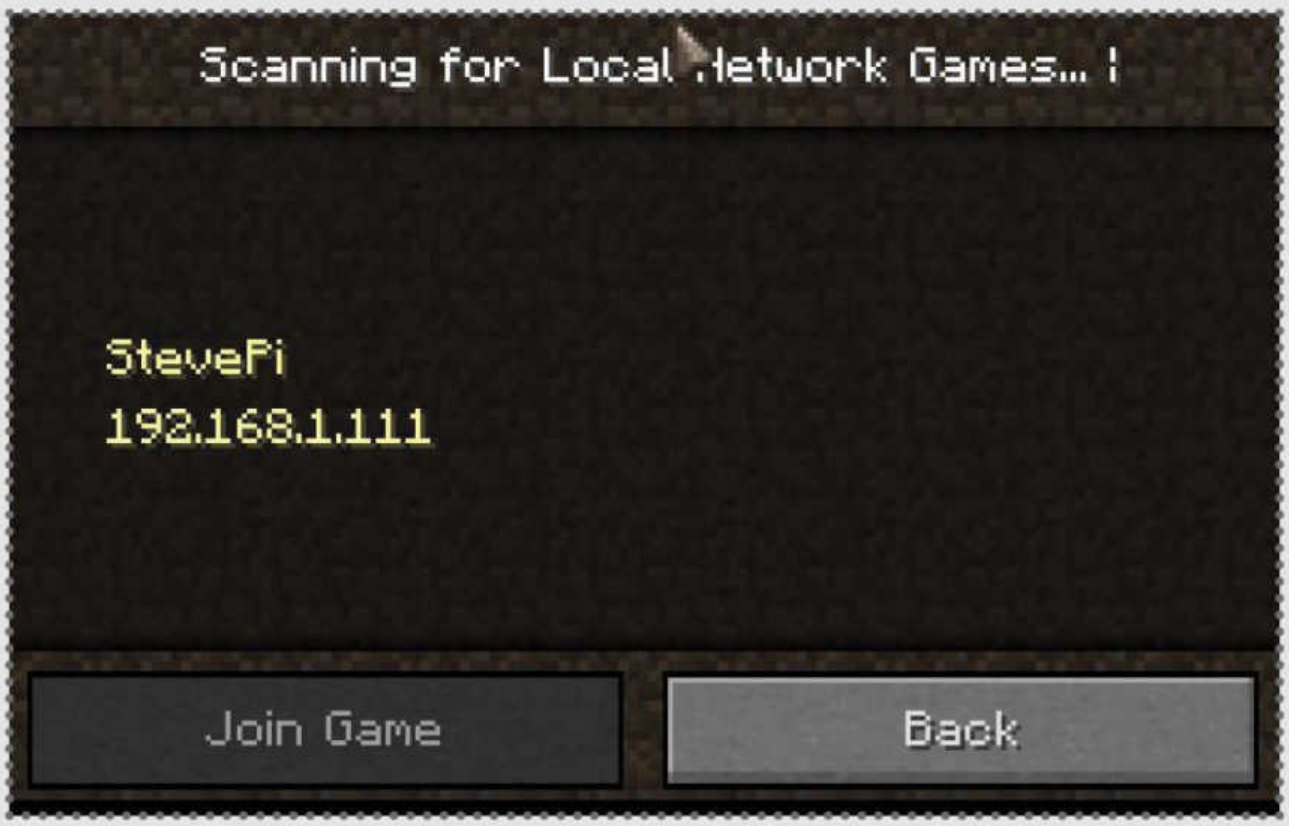
Take your glove and attach the four buttons to the fingers on the glove. There are a number of ways to

do this: glue the button on, sew them in, stick them on with double-sided tape – the choice is up to you. Wires can be hidden or on display, depending on your preferences and how you want the glove to look.

10 Connect the wires

Now you are ready to connect the wires to the Raspberry Pi to enable all four power moves to be used. Take one end of each wire and connect them to the respective pins, as shown below. The other end of the wire is placed into a ground pin. Note that the RPi. GPIO pin numbering system is used here rather than the physical pin number on the board – for example, GPIO pin 17 is physical pin number 11. The following pins are used:

```
GPIO 17, pin 11 on the board
GPIO 18, pin 12 on the board
GPIO 9, pin 21 on the board
GPIO 4, pin 7 on the board
```



Left Next we need to set up a networked multiplayer game, so you can put your power moves to use

11 Set the network up

To interact with other players in the Minecraft world you will need to set up a networked multiplayer game. This is simple and can be achieved using an old router. First, connect each Raspberry Pi via an Ethernet cable to the router and then back to the Ethernet port on each Raspberry Pi. Then turn on your router and wait about 30 seconds for the IP addresses to be assigned. Now load up Minecraft and have one of the players start a new game.

12 Run the game!

After the game has loaded, the other connected player will see an option to "connect to a multiplayer game", usually called Steve. The player selects this option and the networked game will begin. You will be able to see two 'Steves' and the Minecraft chat will inform you that 'Steve has joined the game'. Open Python and the glove code, press F5 to run and you will be able to see the power moves being deployed.

13 Interact in another player's world

You'll notice under the current setup that if the Pi Glove is connected to the game and you use one of your power moves, it will only appear in your Minecraft world and not in the other players. This isn't great, as the power move is supposed to stop the other player! To resolve this, find the IP address of the other Raspberry Pi, then enter this IP address into this line of code: mc = minecraft.Minecraft.create( ). For example, **mc=minecraft.Minecraft.create("192.168.2.234")**, filling the empty brackets with the IP address of the other Raspberry Pi within your game. Remember that

IP address

Most home networks will use IP addresses that start with 192.168.1 and then a number up to 255. An old router will assign these IP addresses automatically. If the router has Wi-Fi then players can also connect to multiplayer using a USB Wi-Fi dongle; you are setting up a LAN (Local Area Network).

Left We'll use a second program to monitor the progress of our game of catch

this is the IP address of the other player's Raspberry Pi and not your IP address.

14 Find your IP addresses

To find the IP address of a Raspberry Pi, load the LX terminal, type **ipconfig** and press Enter – the address will be displayed on the line that begins "int addr:". This is the number that you enter into the **mc = minecraft.Minecraft.create("192.168.2.234")** line. Remember on the Glove Raspberry Pi to enter the IP address of the other player's Raspberry Pi, not yours.

15 Run both programs

No game would be complete without some healthy competition and strategy. A second program is deployed by the other player which tracks and registers if they catch or hit you. The program checks the block that they have hit and compares it to the player's location. Download it now from **https://bit.ly/1OhIKf5**.

16 Test for hits

To check if the other player has hit you, run the second program on the Raspberry Pi of the player who is doing the chasing. The program basically finds the other 'glove' player's current position and stores it in a variable. It then compares the position that you hit with your sword, recording and storing this too. The program then compares the values – if they match then you have hit the other player and have won the game. If not, get ready for a tirade of power moves. Note that in order to monitor where the other player is, you must set the code line mc1 = minecraft.Minecraft.create( ) to the IP address of the Glove Raspberry Pi; for example, **mc1 = minecraft.Minecraft.create("192.168.1.251")**.

17 Game on

Now you are ready to play, check again that the IP addresses are set for the other Raspberry Pi and not your own. Build a new Minecraft world and start a new game on the Raspberry Pi with the player who is chasing. When loaded, the glove player joins the multiplayer game – this will be called Steve (see Step 11). When loaded, you should see both players in the world. Then run the 'Pi Glove power moves' program, and on the other Pi run the 'You hit me program'. Don't forget to set the IP addresses to each other Raspberry Pi. Once set up, you can modify the power moves, use different blocks and add new moves. You could create a timer and a scoring system to track which player can survive the longest. If you are feeling adventurous, you may want to make another Power Glove, one for each player. Have fun!

"Then run the 'Pi Glove power moves' program, and on the other Pi run the 'You hit me program'. Don't forget to set the IP addresses to each other Raspberry Pi"

The Code PIGLOVEPOWERMOVES.PY

```
import time
from mcpi import minecraft

mc = minecraft.Minecraft.create("192.168.1.211")
#Replace with the other players IP address

import RPi.GPIO as GPIO

#Set up the GPIO Pins
GPIO.setmode(GPIO.BCM)

#Sets the pin to high
GPIO.cleanup()
GPIO.setup(17, GPIO.IN, GPIO.PUD_UP)
#11 on the BOARD GREEN Firewall
GPIO.setup(18, GPIO.IN, GPIO.PUD_UP)
#12 on the BOARD BROWN Lava
GPIO.setup(9, GPIO.IN, GPIO.PUD_UP)
#21 on the BOARD BLUE Mega Jump
GPIO.setup(4, GPIO.IN, GPIO.PUD_UP)
#7 on the BOARD ORANGE Puddle
GPIO.setwarnings(False)    #switch off other ports

#Builds a wall of TNT which if the player hits will explode
def Firewall():
    mc.postToChat("Firewall Placed")
    TNT = 46,1
    x, y, z = mc.player.getPos()
    mc.setBlocks(x-6, y, z+2, x+6, y+10, z+3, TNT)
    time.sleep(1)

#Lays Lava to slow down the other player
def Lay_Lava():
```

The Code PIGLOVEPOWERMOVES.PY (Cont.)

```
    Lava = 10
    check = 1
    mc.postToChat("Lava Deployed")
    while check == 1:
        time.sleep(0.2  )
        x, y, z = mc.player.getPos()
        mc.setBlock(x-1, y, z, Lava)
        check = 0

#Peforms a Mega Jump to lose players
def Mega_Jump():
    time.sleep(0.1)
    mc.postToChat("Mega-Jump")
    x, y, z = mc.player.getPos()
    mc.player.setPos(x, y+15, z)
    time.sleep(1)
```

```
#Creates a Puddle to slow down your player
def Mega_Water_Puddle():
    mc.postToChat("Mega Water Puddle")
    time.sleep(0.2)
    WATER = 9
    x, y, z = mc.player.getPos()
    mc.setBlocks(x-5, y, z-4, x-1, y, z+4, WATER)
    time.sleep(1)
#Main code to run
try:
    lava_check = 0
    mc.postToChat("Minecraft Power Glove Enabled")
    while True:
        if GPIO.input(17) == 0:
            Firewall()
        if GPIO.input(18) == 0: #needs to stop
```

The Code PIGLOVEPOWERMOVES.PY (Cont.)

```
            Lay_Lava()

            #GPIO.output(18, GPIO.LOW)
        if GPIO.input(9) == 0:
            Mega_Jump()
        if GPIO.input(4) == 0:
            Mega_Water_Puddle()

except:
    print “Error”
```

The Code YOUWEREHIT.PY

```
import time
from mcpi import minecraft

mc1 = minecraft.Minecraft.create("192.168.1.245")
#The other players IP address goes here

mc = minecraft.Minecraft.create()
mc.postToChat("###Here I come")
Hit = 1

while Hit == 1:

#Find the block stood on
  stood_x, stood_y, stood_z = mc1.player.getTilePos()
  time.sleep(3)

  blockHits = mc.events.pollBlockHits()
  if blockHits:
    for blockHit in blockHits:
      if stood_z == blockHit.pos.z and stood_y == blockHit.pos.y+1:
        mc.postToChat("I got you")
        time.sleep(2)
        mc.postToChat("###GAME OVER###")
        time.sleep(1)
        Hit = 0
        mc.postToChat("###Restart Hit Code")
```

Monitor your local network with NagiosPi

Embrace the power of Nagios to keep an eye on servers, switches and applications on your network

Is your PC offline? Has your Linux box stopped serving Minecraft or Counter-Strike? If you're out of the house, or even the country, there is no real way of knowing without trying to log in – something you probably won't be able to do without being on the premises (unless you're using remote desktop software).

A far better way would be to simply receive notifications when your network devices are knocked offline, and this is why we turn to NagiosPi, a Raspberry Pi-built version of the popular open source network monitoring tool.

NagiosPi is available as a full image ready to be written to SD card, with the real configuration taking place once it's up and running. Let's get started.

THE PROJECT ESSENTIALS

NagiosPi
http://bit.ly/1KY38XF

Win32 Disk Imager
http://bit.ly/L8JdYG

Disk Utility
http://bit.ly/1Lec9r5

Internet connection

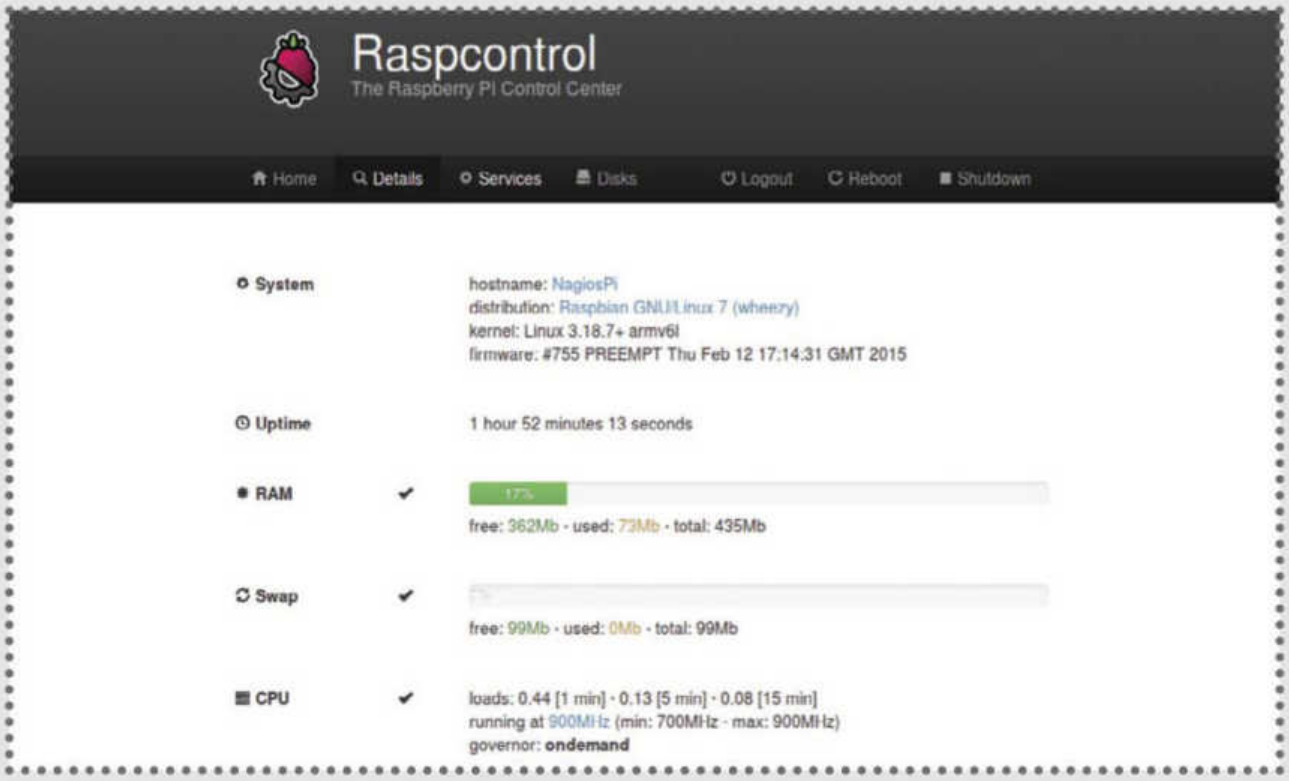
4GB+ SD card

01 Download and write NagiosPi to SD

Windows users should write the extracted contents of the NagiosPi_v2.0.zip file to a formatted SD card using Win32 Disk Imager. Linux desktop users can use Disk Utility or the command line (**http://bit.ly/1z36sp8**). With the image written to SD, safely eject the card and insert it into your Pi before booting.

02 Log in to NagiosPi

As with most Pi projects, you'll probably want to operate via SSH, so check your router's list of connected devices to find the IP address and connect. You can also use a keyboard and monitor connected to your Raspberry Pi. The default username and password for NagiosPi is pi/raspberry.



Left The control centre gives you a handy at-a-glance status update

03 Expand the filesystem

Before proceeding, run **sudo raspi-config**. You'll need to select the first option, Expand Filesystem, and wait a moment as the filesystem is expanded to the full size of the SD card. Once done, select Change User Password to add some security to your NagiosPi, then select Finish and reboot.

04 Open NagiosPi in your browser

With the Pi rebooted, you'll be able to open the NagiosPi web console in your browser. Visit http://[your.IP.address.here] to see the available options. Here you'll spot a menu of links in the top-left corner, each accompanied with the username and password to sign in. Start with RaspControl.

05 Monitoring your NagiosPi box

In the RaspControl section you'll get a flavour of just what Nagios can do. On the home screen you'll see general hardware information such as connectivity and system status, and as you flick through Details,

Services and Disks you'll see what level of monitoring is possible.

06 View host status in NagiosPi

Next, go to Nagios and pick Hosts. Here you will see the current status for the configured hosts, which is a combination of items detected on your local network and preset entities. Look for Current Network Status in the upper-left area of the console; just below this you will find alternate views.

07 Adding a host to monitor

Open NConf to add the server you wish to monitor, using the Hosts – Add button to input the device hostname, IP address and alias. Click Submit when done, then switch to Services – Add, where you can assign a name and check command (such as check_ping) to monitor.

08 Create your configuration file

Each check must be set up individually. Some require the installation of NRPE (Nagios Remote Plugin Executor) on remote devices to interrogate and present full system details, but this isn't necessary for basic things like ping. When you're done, click Submit, then Generate Nagios Config. Following this, select Deploy.

09 Monitor your server

In the NagiosPi window, select Services to see currently monitored servers and devices. For each device, there will be extra info that you drill down into by clicking under Actions. We've shown you the basics of NagiosPi – investigation will demonstrate just how powerful it is!

"Each check must be set up individually. Some require the installation of NRPE on remote devices to interrogate and present full system details, but this isn't necessary for basic things like ping"

FAQ What is PiJuice?

PiJuice is a new power option for the Pi. What is it and what makes it special?

So, the PiJuice – has the Raspberry Pi Foundation been dabbling drinks licensing recently, then?
No, it is a physical product for the Raspberry Pi – Juice is just a descriptive word for what it does, sort of.

Okay then, what is the PiJuice?
The PiJuice is a way to make the Raspberry Pi more portable – the juice bit comes from the colloquial for power or energy, which is one of the main components of this system that enables a portable Raspberry Pi.

What do you mean by power or energy?
In this case, a battery with electrical power for the Raspberry Pi. Portable batteries for mobile phone charging exist, sure, but none are optimised for use with the Raspberry Pi.

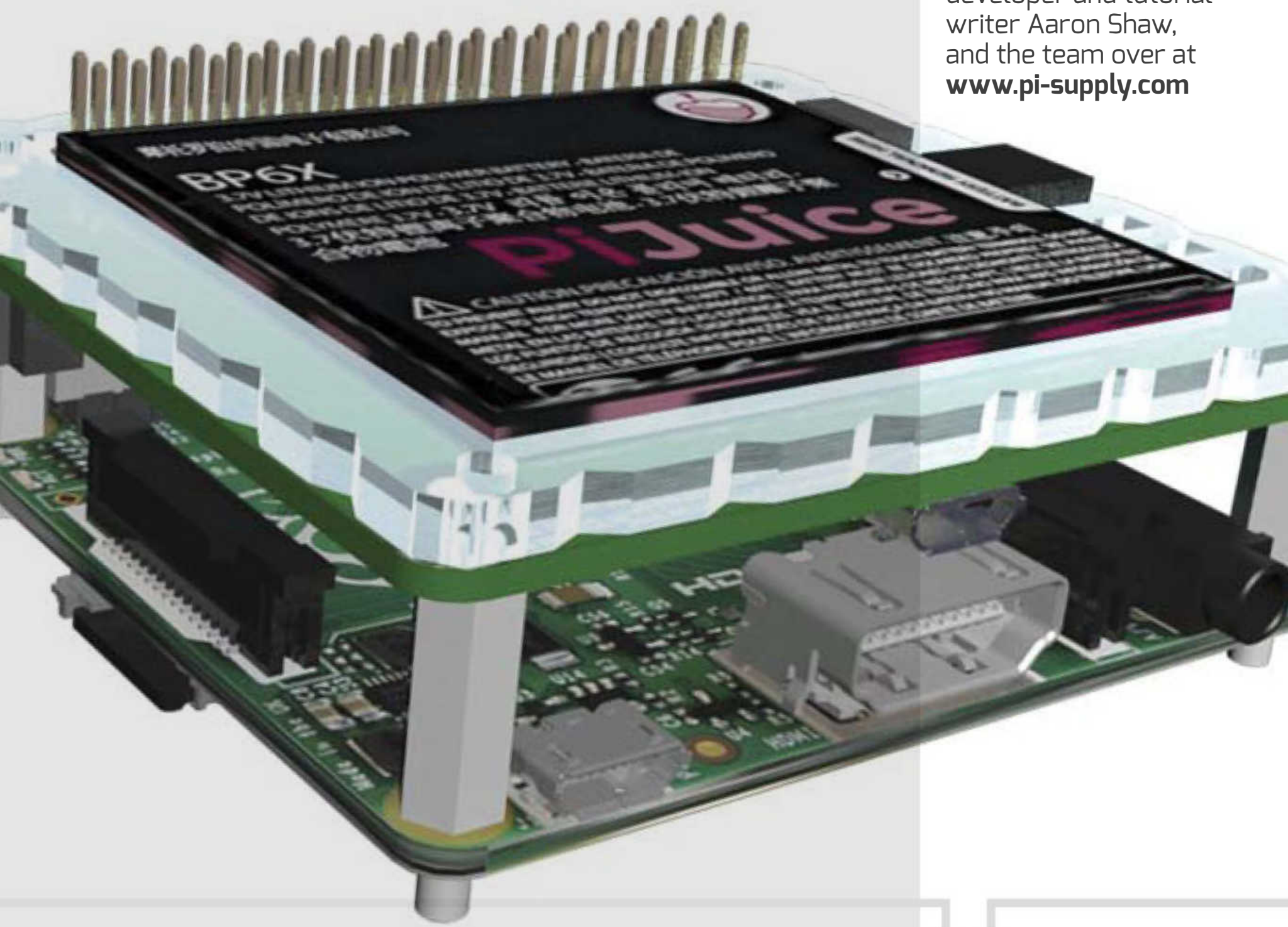
How is the PiJuice optimised for the Raspberry Pi?
Well, it uses a rechargeable lithium battery that is the correct size to keep flush with the Raspberry Pi and not make it too bulky, and it offers uninterrupted power supply, meaning you have a more consistent flow of power to

"It uses a rechargeable lithium battery that is the correct size to keep flush with the Raspberry Pi and not make it too bulky"

the Pi when this is plugged in compared to some of the alternative portable chargers available.

You said this power bit was only one of the main components – so what else does it include?

To be truly portable, you need to have a bit more than just powering the device. You also need to have access to a clock that's off the grid, and this has a real time clock built in. To extend the battery life it needs to have power management options; the PiJuice has both hardware and software solutions to aid this. It's also very low profile, making it more flexible when you are looking for a Raspberry Pi case.



Below PiJuice was created by Pi developer and tutorial writer Aaron Shaw, and the team over at **www.pi-supply.com**

What about a screen? Isn't that important for portability with the Pi?
It can be, which is why you can get a special 'header' for the GPIO pins that enable you to then add other components on top of the PiJuice ones and use both at the same time. In this case, one of the PiTFT screens or a larger screen for bigger projects can both be used.

Is this an official Raspberry Pi thing?
No, this is a separate hardware developer, but it's caught the eye of the Raspberry Pi Foundation because it does plug a gap in the market in terms of powering the Raspberry Pi properly while being portable as well.

Does it do anything else special?
As well as having special kits for makers and people looking to get into making (including a DIY camera and games console), there is also a special version which will let you run and/or charge the Pi off of solar power which could come in pretty useful.

Solar power? How would that be useful for the PiJuice?
Well if you're using it for an outdoors project, you can have it constantly powered without requiring much maintenance, or constant charging. It's a very neat idea!

So how much does it cost and where can I get it from?
The price starts at £25, which is pretty cheap, and then goes up to £80 for a full kit that you can start working with straight away. It was recently (and successfully) Kickstarted, so you can pre-order a kit now that should be begin shipping out to you in the coming months.

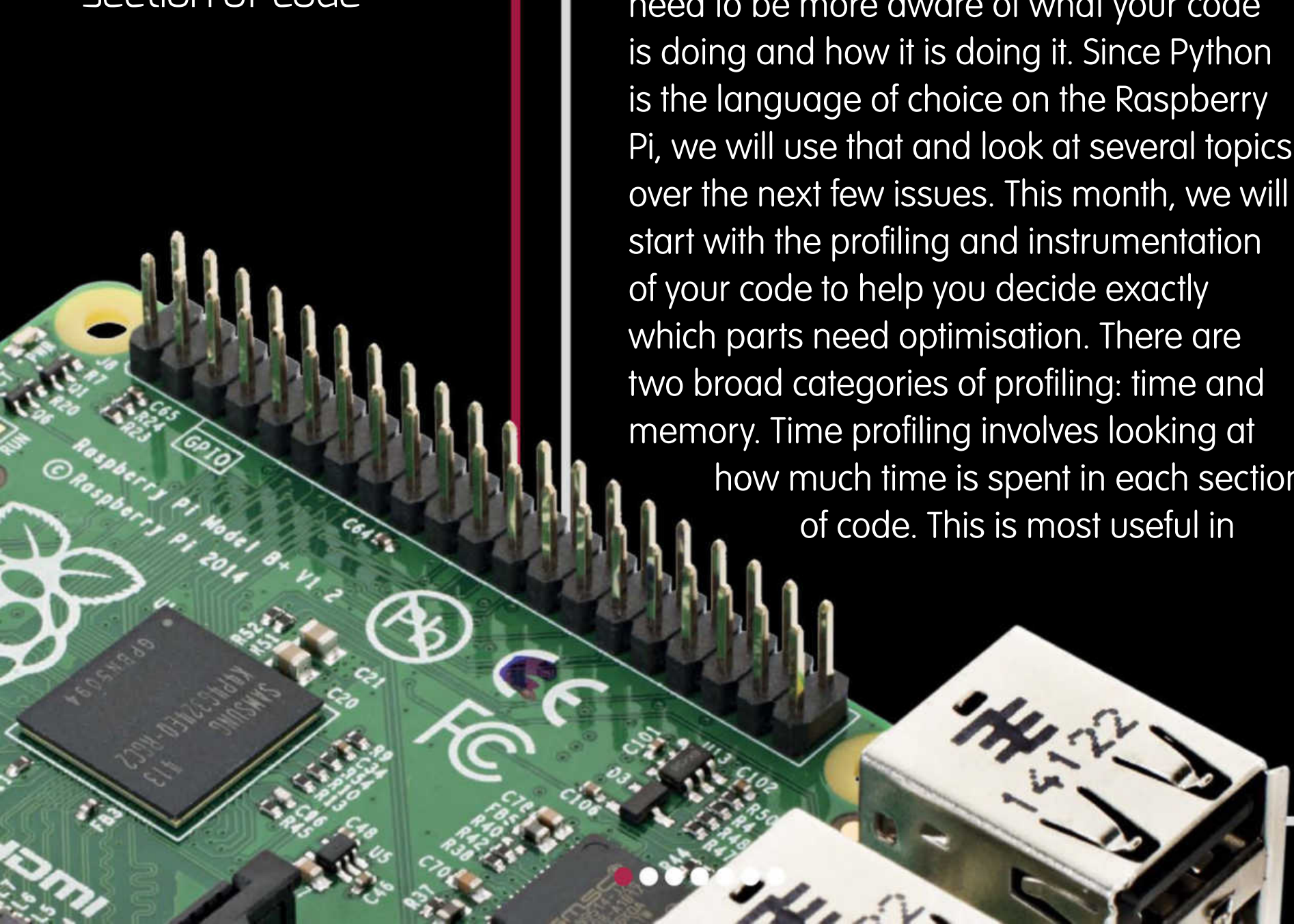
"There is also a special version which will let you run and/or charge the Pi off of solar power"

Profiling Python code to calculate efficiency

To maximise your Raspberry Pi, you need to use profiling to figure out exactly where the problems are

"Time profiling involves looking at how much time is spent in each section of code"

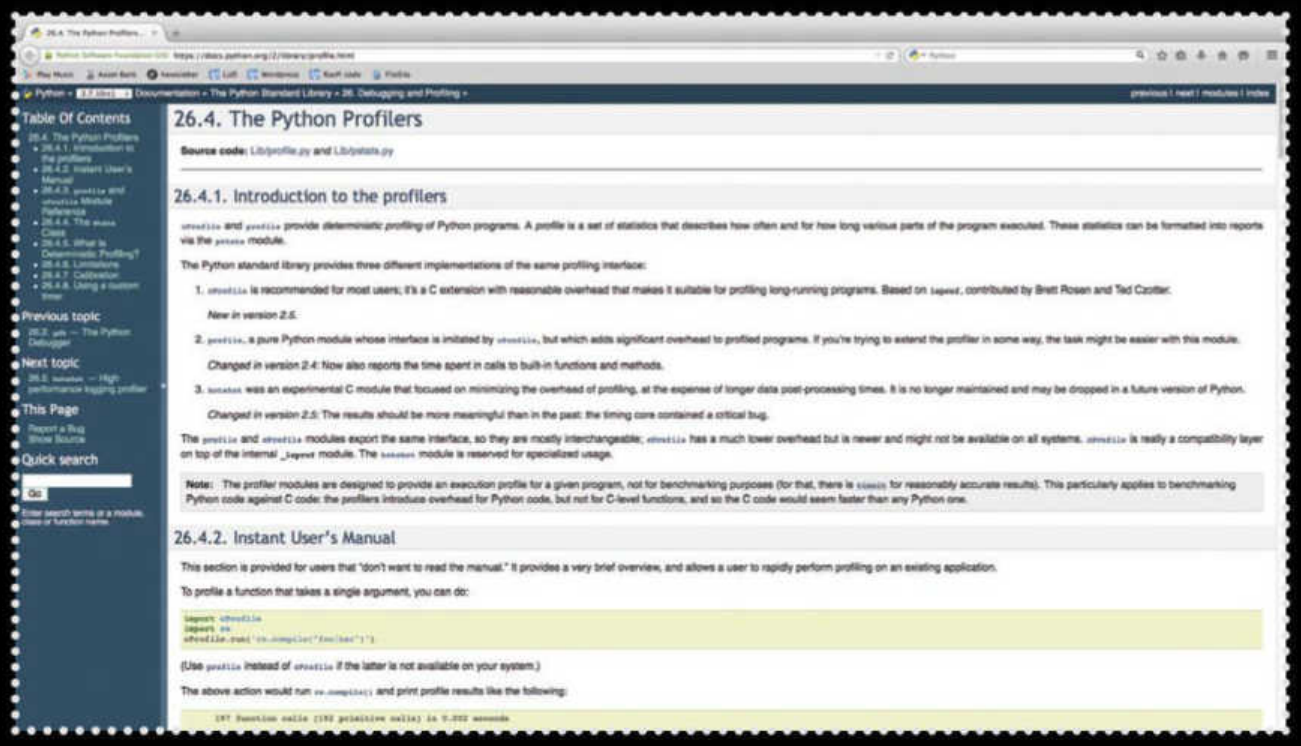
One of the problems when writing code on a Raspberry Pi is the fact that your computational resources are limited, at least by modern standards. This means that you, as a programmer, need to be more aware of what your code is doing and how it is doing it. Since Python is the language of choice on the Raspberry Pi, we will use that and look at several topics over the next few issues. This month, we will start with the profiling and instrumentation of your code to help you decide exactly which parts need optimisation. There are two broad categories of profiling: time and memory. Time profiling involves looking at how much time is spent in each section of code. This is most useful in



trying to improve the overall algorithm you are using and the specific way that you have implemented it. Memory profiling is used to analyse how efficiently you are using memory and to potentially find areas where you are wasting memory through leaks or simply inefficiencies. There are also two different methods of doing profiling: deterministic and probabilistic. Deterministic profiling tracks the execution of every instruction within your program. The advantage of this method is that you get perfectly correct results. The disadvantage is that it introduces a huge amount of overhead to your program. This means that you cannot do any benchmarking at the same time as you do profiling. The extra overhead also means that it may not be practical to use deterministic profiling on a program that needs to run for a long period of time. The second method of profiling is probabilistic profiling. This method essentially samples your program at some regular interval to see what instruction it is executing. The advantage to this method is that it introduces almost no overhead, and so is ideal for long-running programs. The problem is that you are only collecting statistical data about how often each instruction is being executed. This means that you may actually miss some information. You should end up collecting data on the most heavily-used instructions, on average, so this should not cause any problems in most cases.

The simplest form of time profiling is to simply use the 'time' function. You can record a start time and an end time around a chunk of code to see how long the given section takes to run. While this is easy, it is also relatively coarse and takes up quite a bit of programmer time. The Python standard library itself includes two profiling modules: profile and cProfile. Profile is a pure

"Memory profiling is used to analyse how efficiently you are using memory and potentially find areas where you are wasting memory through leaks or simply inefficiencies"



Left Go to section 26 of the Python documentation website (Debugging and Profiling) and select your Python version to learn more

Python profiling module that offers deterministic time information. The overhead for this module is very high, but because it is pure Python it is easy to extend and add in any extra functionality that you may need. If you don't need that much control, you can use the module cProfile instead. This profiler is actually written in C and imported into Python. This results in quite a lower amount of overhead. In both cases, after importing the relevant module, you can use the function run() to profile a given function. By definition, this means that your code needs to be packaged as a callable function that can be handed in to run(). The other option available is to include a call to cProfile on the command line. It would look like:

```
python -m cProfile myscript.py
```

In this way, you can profile an entire Python script file rather than just a given function. The default output you get is a summary line giving the total number of functions calls and how long the entire process took. Below this, you are given a breakdown of each function with the following columns:

ncalls – number of calls
tottime – total time in each function
percall – tottime divided by ncalls
cumtime – cumulative time spent in each function plus all sub-functions
percall – cumtime divided by number of primitive calls
filename:lineno(function) - location data for each function call

Another parameter for the run() function takes a filename in which to save the raw profiling code. You can then use this raw data to do more complex analysis. For example, you can use the stats class from the pstats module. You can then apply some different sorting schemes or do some filtering of the data. You can also print out how many callers or callees each function has.

For memory profiling, the most common option is to use a third-party module called memory_profiler. This module is available through pip, so you can install it with the command:

```
pip install memory_profiler
```

If you want to see the memory usage for an entire script, you can use it in a similar way to using cProfile:

```
python -m memory_profiler
myscript.py
```

The output from memory_profiler is given by the line, rather than by the function. For each line, it will print out the current memory usage, the increment in memory usage from the previous line and the executable

"Memory profiling is used to analyse how efficiently you are using memory and potentially find areas where you are wasting memory through leaks or simply inefficiencies"

contents of the given line. If you want to narrow down your area of interest to a single function, you can import a function decorator to do memory profiling one function at a time. To use it, you would use:

```
from memory_profiler import profile
@profile
def my_func():
```

It is important to note that the cProfile module also includes a decorator named profile, so you won't be able to use both at the same time. Memory_profiler also includes an executable called mprof that can be used to profile external scripts or codes. To record data, use:

```
mprof run myscript.py
```

You can then get a time plot of memory usage with the following command:

```
mprof plot
```

There are several other functions available through mprof. Now that you know which parts of your code are in need of attention, be sure to check back in next issue. We will look at optimisation tips and tricks that may apply to your specific problem. If not, they may point in directions that you should consider further.

"The cProfile module also includes a decorator named profile, so you won't be able to use both at the same time"

The Code PROFILING

```
# The following file can be used whenever you use one of the available decorators
# from memory_profiler or cProfile. You would run it with a command like:
#     python -m memory_profiler myscript.py
```

myscript.py

```
@profile
def my_func(x):
    if x == 0:
        return 1
    elif x == 1:
        return 1
    else:
    return my_func(x-1)
```

my_func(10)

```
# The following file can be used to explicitly use the run functions from the
# various profilers:
```

myscript2.py

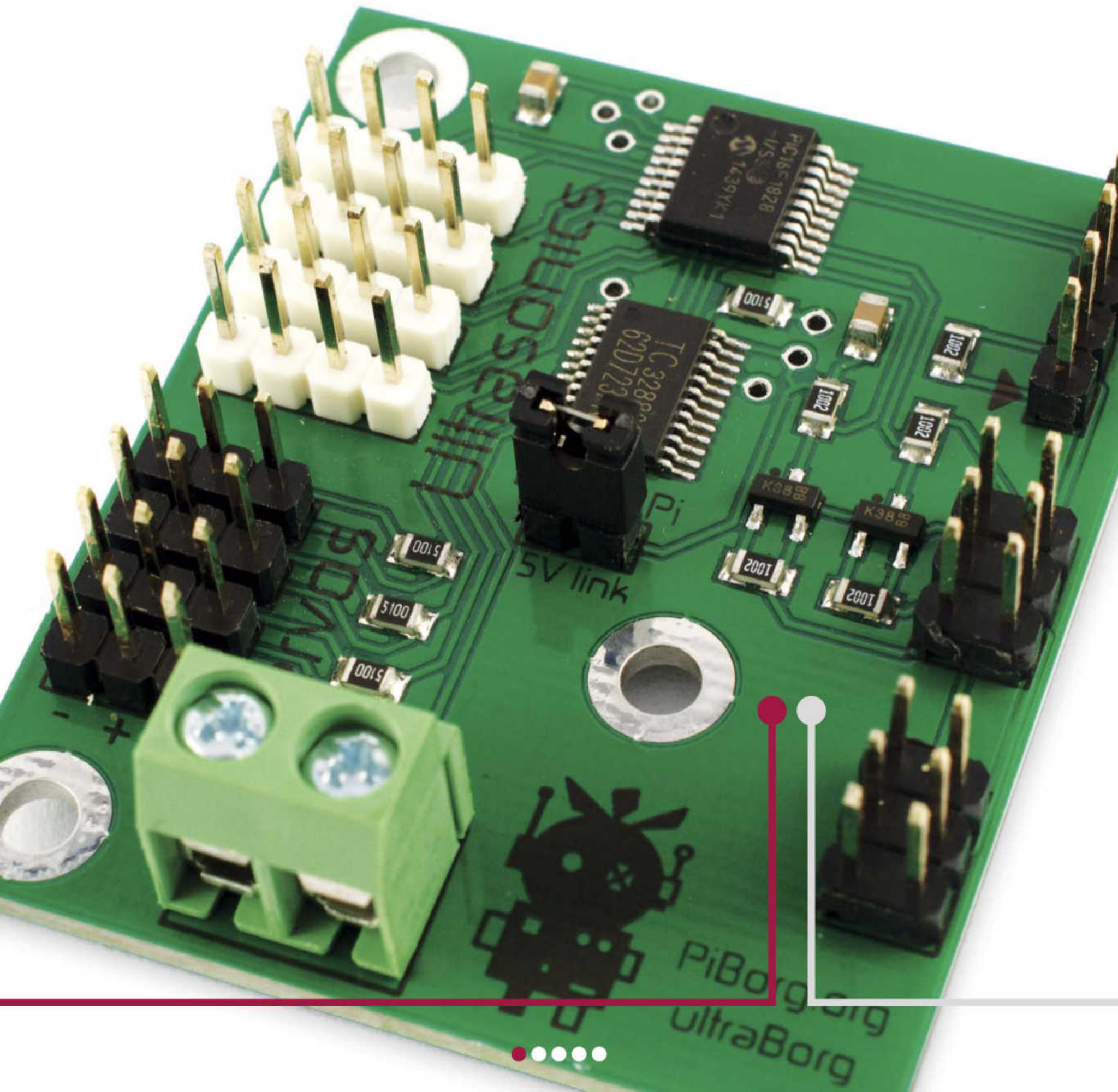
```
import memory_profiler as mp

def my_func(x):
    if x == 0:
        return 1
    elif x == 1:
        return 1
    else:
    return my_func(x-1)
```

The Code PROFILING

```
mp.run("my_func(10)")
------------

# The output would be:
------------
   12 function calls (3 primitive calls) in 0.000 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall  filename:lineno(function)
        1    0.000    0.000    0.000    0.000  <string>:1(<module>)
     10/1    0.000    0.000    0.000    0.000  myscript2.py:3(my_func)
        1    0.000    0.000    0.000    0.000  {method 'disable' of '_lsprof.
                                               Profiler' objects}
```

PiBorg UltraBorg

Designed for Raspberry Pi robotics projects, does the expandable UltraBorg deliver on its promises?



Designed primarily for the Raspberry Pi yet compatible with any microcomputer or microcontroller that can speak inter-integrated circuit (I2C) – including the popular Arduino and compatible boards – PiBorg's UltraBorg is built with one task in mind: robotics.

The majority of hobbyist-level robotics projects use two major components: servos, controlled via pulse-width modulation (PWM) to move the robotic parts, and ultrasonic sensors, which send out a pulse of inaudible sound and listen back for the echo to measure distance to solid objects. Combined, that's enough to build everything from a simple two-wheel robot to a more complex robotic arm, although computer vision, aided by devices like the Raspberry Pi Camera Module, can help with accurate positioning for the latter.

While it can't help you with the camera side of things, the UltraBorg is positioned as the ultimate one-

Left PiBorg is behind the caravan-towing DoodleBorg (see **http://bit.ly/1L1qgmL**) and boards like the XLoBorg motion and direction sensor

TECH SPECS

Type
Add-On Board (Non-HAT)

Processor
PIC16F1824-I/SL

Driver
Toshiba 16-bit PWM IC TC62D723FNG

Outputs
4x 5V PWM (Servo-compatible)

Inputs
4x 5V Ultrasonic Sensor

Power Supply
5V required for servo use, not supplied

Price
From £15.99 ($25 approx)

Available from
http://piborg.org

stop device for budding roboticists. The board is split into two four-channel groups: the first group provides input support for four ultrasonic sensor modules; the second group provides output support for four servos or other PWM-controlled motors. All told, that's enough for a reasonably proficient robot, but if you need more it's possible to daisy-chain multiple UltraBorgs together to gain additional input and output channels, a handy feature that is tempered only by the £15.99 asking price per board.

The UltraBorg includes an on-board processor, a PIC, which provides for real-time sensing and control while taking the pressure off the Pi – more of an issue for the single-core Model B+ and Model A+ than the quad-core Raspberry Pi 2 – while a Toshiba PWM controller provides 16-bit precision, an improvement on the 8- or 12-bit of its rivals. You don't need to know how to program a PIC to use the device as the chip arrives pre-

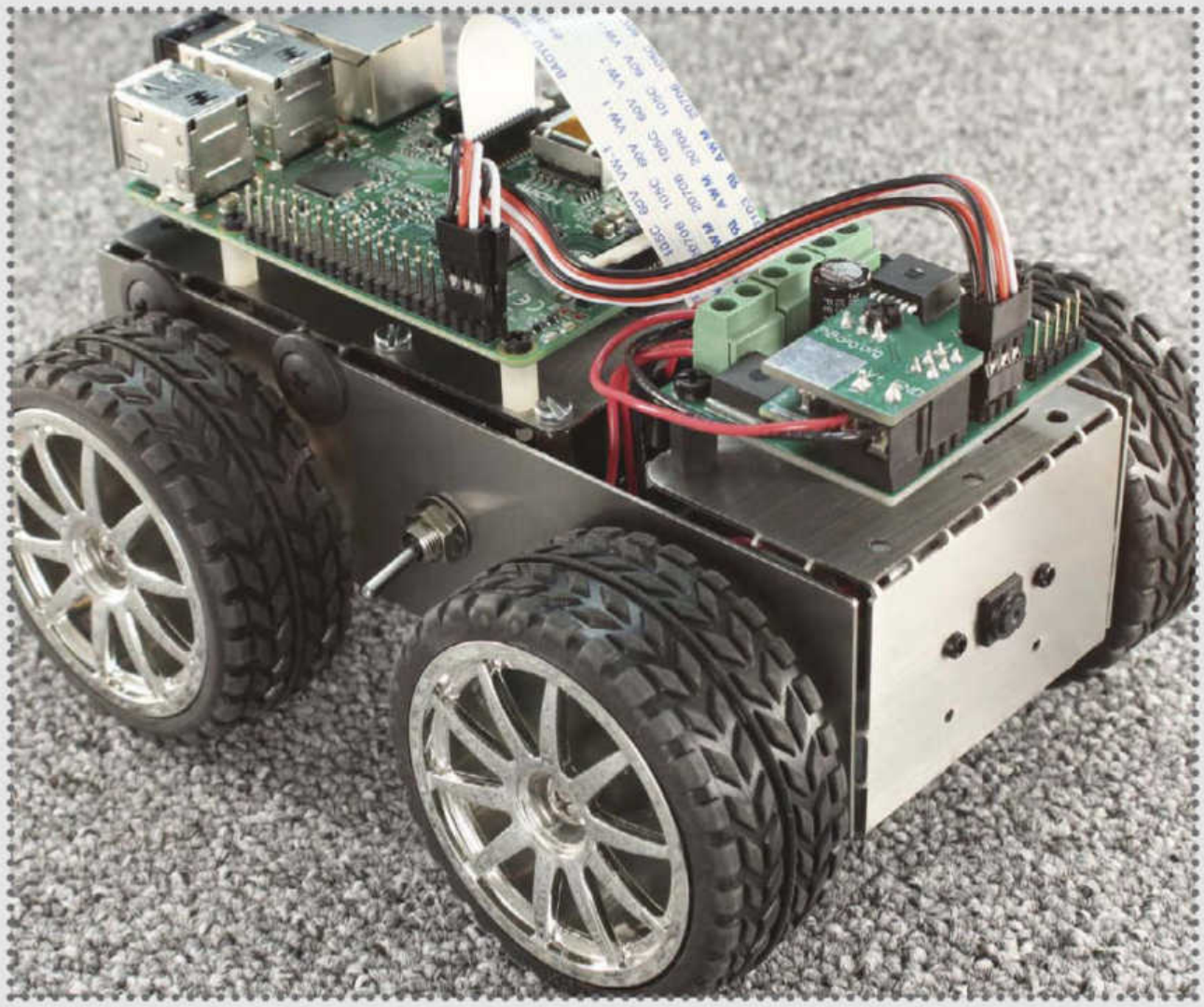
Above This Toshiba processor handles the PWM for the UltraBorg-connected motors and servos

programmed and sample code is also provided for user-level interaction.

Installation of the UltraBorg is simple, taking up the first six pins of the GPIO header, although the use of multiple ultrasonic sensors or any number of servos will require a separate 5V power supply, which is not provided with the board. This connects to a screw terminal and can also supply power to the Pi itself, along with any additional UltraBorg boards, to make wiring as easy as possible to conduct.

A particularly handy feature of the UltraBorg is the electrically erasable, programmable read-only memory (EEPROM) located within the PIC process, which is used to store start-up and limit values for the PWM outputs. Using this you can set a custom zero-point for your particular servos, while also limiting their minimum and maximum rotation points. When power is cut and reapplied, the servos will automatically return to the zero

"A particularly handy feature of the UltraBorg is the electrically erasable, programmable read-only memory (EEPROM) located within the PIC process"



Left PiBorg also make some truly excellent robots, like this armour-plated 4Borg that costs just £99

you have set. For protecting your build against damage from improper values or bad defaults, it's a must-have.

Although the UltraBorg isn't a Hardware Attached on Top (HAT) standard device, it has been designed with piggyback mounting in mind. Screw holes on the board itself are positioned to enable it to be attached above or below any model of Raspberry Pi bar the Compute Module or the very original Model B. These same holes can also be used to mount the device elsewhere if you are sacrificing footprint for thinness or using it with a different microcomputer or microcontroller, a job made easier thanks to the small size and weight of the board.

PiBorg hasn't just launched the device blindly either. The Pi-centric company has designed a range of accessories, including a BattBorg battery power supply, a 90-degree mounting bracket for the ultrasonic sensors and bundles which include quality servo motors, sensors, and all the connecting cables.

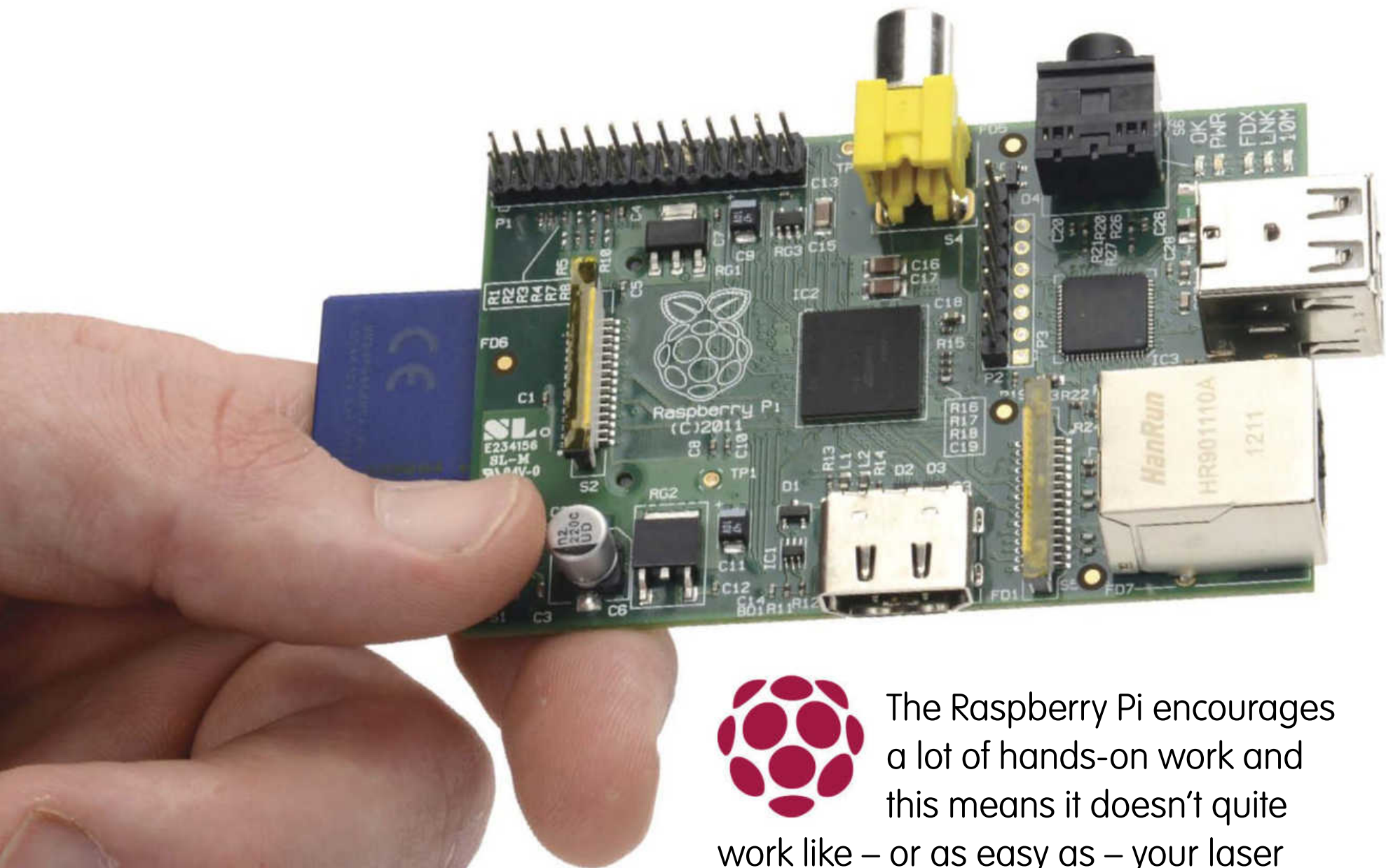
THE VERDICT

A marvellous device, with a compact size that belies its power. Combining input and output onto one board means it's a literal one-stop-shop for simple robotics projects, and daisy-chaining support makes extending it simple. Sadly, it is quite expensive compared to the 16-channel competition out there.

Talking Pi

Join the conversation at...

@linuxusermag | Linux User & Developer | RasPi@imagine-publishing.co.uk



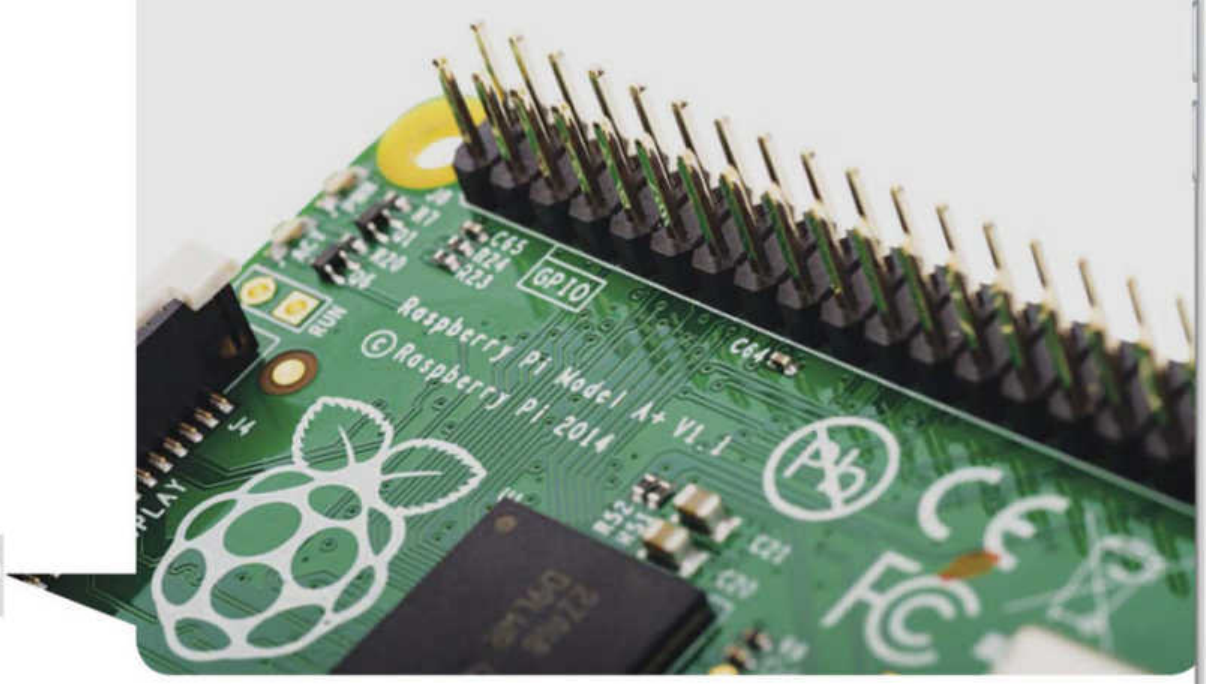
The Raspberry Pi encourages a lot of hands-on work and this means it doesn't quite work like – or as easy as – your laser focus-tested smartphone interface, or even a normal computer. So we're answering your burning Raspberry Pi questions each issue – get in touch with us on Twitter, Facebook or by email.

What is the maximum rated current that can be drawn on the 3V and 5V lines respectively when powering my external projects?
Carl via email

Thanks for writing in, Carl – here are some stats for you:

3V3 – There is a 50mA maximum current draw for all of the 3V3 pins. There's a limit of approximately 12-15mA per pin (figures aren't certain!) – so basically, spread your 50mA allowance over no more than three or four pins to be safe, depending on the draw of each pin. With low power draw – ie no Ethernet, no HDMI, etc – some users think that the overall limit could increase by as much as 100-200mA, giving your 150/250mA to play with. This has not been tested, however!

5V – With the 5V rail, your maximum current draw is the USB input minus the current power draw. So for an old model B that would be 1000 – 700 = 300mA max. The B+ and 2B would be 1000 – 650 = 350mA max. Hope that helps!



Keep up with the latest Raspberry Pi news by following @LinuxUserMag on Twitter. Search for the hashtag #RasPiMag

JUST A SCORE
WHAT'S YOUR JUST A SCORE?

Have you heard of Just A Score? It's a new, completely free app that gives you all the latest review scores. You can score anything in the world, like and share scores, follow scorers for your favourite topics and much more. And it's really good fun!

Pi Wars was great! I miss it already. When is the next party?
Amy via email

You're in luck, Amy – the next party is the Birthday Weekend on 5-6 March, held in Cambridge at the same place (University of Cambridge Computer Laboratory). The Raspberry Pi turns 4 years old this year, so we're all getting together to celebrate with games, workshops, code and cake. See you there!

I like Minecraft, but can I play other games on my Pi?
Dom via email

Hi Dom – you sure can! Nothing like Elite Dangerous, unfortunately, but you can run games like the Oolite clone, for example, or OpenArena. The latest Raspbian release includes an experimental OpenGL driver, which means 3D games (within the reach of your Pi's CPU and RAM) can now be played.

Next issue

Get inspired **Expert advice** **Easy-to-follow guides**

Code a simple Synth

Get this issue's source code at:

www.linuxuser.co.uk/raspicode